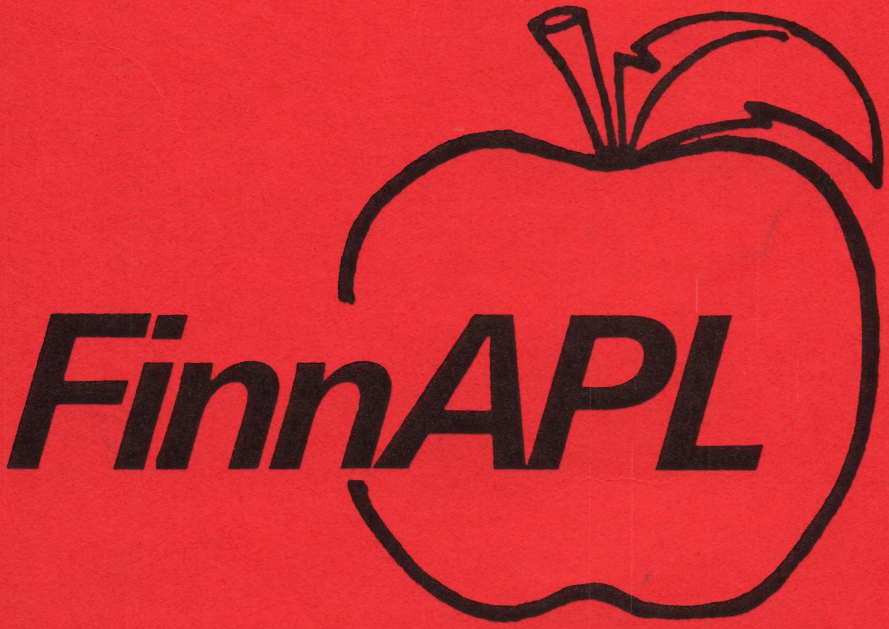
FINNISH APL ASSOCIATION

FINNAPL UTILITY LIBRARY

FIRST EDITION

APRIL 15. 1985



Helsinki 1985

INTRODUCTION

"APL Special" is an auxiliary function library based on the "Special" workspace of The Finnish APL Association FinnAPL. The 101 most important functions in this workspace are documented here. In accordance with the widely accepted standards, the last letter of the name of each function is underlined. Some other alterations have been made on the functions (of the old versions of the workspace Special). However, none of these alterations have affected the operations of the functions.

Each function documentation contains a few examples which give some ideas in regard to the use of the functions. The examples are laid out as they can be seen on the terminal.

The Auxiliary Functions are designed to be extensions of APL primitives. These functions may be more effective than the original primitives, or they may perpetrate more complex tasks. In case of dyadic functions, the relationship between the left and right argument is usually the same as for the primitives. The functions whose results depend on the setting of system variable Index Origin will be noted separately.

The function descriptions are given in alphabetic order within each group.

TABLE OF CONTENTS

Name: AP - Arithmetic progression

Syntax: Z←AP X

Description:

X: Numeric vector of three elements
Z: Arithmetic progression from X[1] to X[2] step X[3].
In absence of the third element, the default value is either ¯1 or 1.

Any number of elements is allowed in the argument but only the three first ones are taken into account.

Example:

```
      AP 2 12 1.5
2 3.5 5 6.5 8 9.5 11
      AP 6 2
6 5 4 3 2
```

Function listing:

```
    ∇ Z←AP X;E
[1]  ⍝⍝ARITHMETIC PROGRESSION
[2]  ⍝FROM X[1] TO X[2] STEP X[3]
[3]   Z←((1↑X)-Z×⎕IO)+Z×⍳1+⌊E÷Z←1↑2↓X,×E←-/2↑X
[4]  ⍝840731 11.52
    ∇
```

Name: AXIS - Horizontal axis indicating column positions

Syntax: Z←AXIS X

Description:

X: Non-negative integer
Z: Matrix of horizontal length X. Indicates the positions of columns in a horizontal axis running from 1 to X(⎕IO=1) or 0 to X-1 (⎕IO=0).

Can be used in written documents etc when it is necessary to locate the positions of certain colums. The result is origin dependent.

Example:

```
      PAGEWIDTH←51
      AXIS PAGEWIDTH
000000000111111111122222222223333333333444444444455
123456789012345678901234567890123456789012345678901
      ⎕IO←0
      AXIS 12
000000000011
012345678901
```

Function listing:

```
    ∇ Z←AXIS X
[1]   ⍝⍝HORIZONTAL AXIS INDICATING COLUMN POSITIONS
[2]   ⍝FROM ⎕IO TO X+⎕IO-1
[3]    Z← 1 0 ⍉((1+⌊10⍟X)⍴10)⊤⍳X
[4]   ⍝840530 09.38
    ∇
```

Name: BI - Test if X is between the limits of Y

Syntax: Z←X BI Y

Description:

X: Numeric array
Y: Two-element vector giving the lower and upper limits of the elements of X
Z: Logical array indicating which elements of X are between Y[1] and Y[2]. Values equal to Y[1] or Y[2] will be counted within the limits.

Example:

```
      ¯2 BI ¯1 1
0
      M←2 3⍴⍳6
      M
1 2 3
4 5 6
      M BI 2 4
0 1 1
1 0 0
```

Function listing:

```
    ∇ Z←X BI Y
[1]   ⍝TEST IF X IS BETWEEN THE LIMITS OF Y
[2]   Z←(X≥1↑Y)^X≤¯1↑Y
[3]   ⍝840530 06.17
    ∇
```

Name: DROUND - Rounds Y to X decimals so that (+/Z)= X ROUND +/Y

Syntax: Z←X DROUND Y

Description:

X: Integer
Y: Numeric array
Z: Argument Y with its elements rounded to X decimals so that the sum of rounded elements equals to the result of rounding the sum of original elements. A negative left argument rounds the elements of Y to 1E¯X.

Example:

```
      7 2⍕2 DROUND 10.102 10.103 10.109 10.102
 10.10  10.11  10.11  10.10
      ¯3 DROUND ⍉⍤3 2⍴1200 3400 7100 5200 12500 100
 1200  3400
 7100  5200
12500   100
 1000  4000
 7000  5000
13000     0
```

Function listing:

```
    ∇ Z←X DROUND Y;P;I;E;S
[1]  ⍝ROUNDS Y TO X DECIMALS SO THAT (+/Z)=X ROUND +
     /Y
[2]   D←⍴Y
[3]   S←⌊0.5+(10*X)×+/Y←,Y
[4]   P←S-+/Z←⌊0.5+E←Y×10*X
[5]   I←P↑⍋Z-E
[6]   Z[I]←Z[I]+×P
[7]   Z←D⍴(10*-X)×Z
[8]  ⍝840801 08.12
    ∇
```

Name: EPS - X ε Y for integers between ⎕IO,9999+⎕IO

Syntax: Z←X EPS Y

Description:

X: Integer scalar or vector with elements between ⎕IO,9999+⎕IO.
Y: Same as X
Z: Logical vector of length X indicating which elements of X are represented in Y.

This function works like the primitive function ε, but is more effective.

Example:

```
      1000 2000 3000 EPS 2000 100 2000 5000
0 1 0
```

Function listing:

```
    ∇ Z←X EPS Y
[1]  ⍝X ε Y FOR INTEGERS BETWEEN ⎕IO,9999+⎕IO
[2]   Z←10000⍴0
[3]   Z[Y]←1
[4]   Z←Z[X]
[5]  ⍝840723 08.37
    ∇
```

Name: EXPND - Expansion vector to expand an array by X

Syntax: Z←X EXPND Y

Description:

X: Non-negative integer vector or scalar
Y: Non-negative integer vector of length X.
   +/Y ↔ length of the axis to be expanded.
Z: Expansion vector which expands an array with
   fields Y by inserting X[I] zeroes or blanks
   (according to the type of the array) after field
   of width Y[I]

A vector argument X is extended to conform with Y.
The function needs the auxiliary function RHO.

Example:

```
      □←B←3 EXPND 1 2
1 0 0 0 1 1 0 0 0
      M←2 2 3⍴⍳6
      B\M
1 0 0 0 2 3 0 0 0
4 0 0 0 5 6 0 0 0

1 0 0 0 2 3 0 0 0
4 0 0 0 5 6 0 0 0
      B←1 3 1 EXPND 2 1 3
      B
1 1 0 1 0 0 0 1 1 1 0
      B⍀6 15⍴'ADDS BLANK ROWS'
ADDS BLANK ROWS
ADDS BLANK ROWS

ADDS BLANK ROWS



ADDS BLANK ROWS
ADDS BLANK ROWS
ADDS BLANK ROWS
```

Function listing:

```
    ∇ Z←X EXPND Y
[1]  ⍝⍝EXPANSION VECTOR TO EXPAND AN ARRAY BY X
[2]  ⍝VECTOR Z  INSERTS X[I] ZEROES AFTER FIELD OF Y
      [I]
[3]  ⍝NEEDS RHO
[4]   Z←(,Y,[□IO+0.1] X) RHO(2×⍴Y)⍴ 1 0
[5]  ⍝841119 15.09
    ∇
```

Name: FITS - Number of characters that fit into space without dividing words

Syntax: Z←X FITS Y

Description:

X: Positive integer
Y: Character vector
Z: Number of characters that fit into a space of width X so that the words are not divided. The character string is given in vector Y.

Blank works as a separator.

Example:

```
      TEXT←'THIS TEXT FITS INTO A GIVEN SPACE-RANGE?'
      ⍴TEXT
40
      ⎕←A←35 FITS TEXT
27
      A↑TEXT
THIS TEXT FITS INTO A GIVEN
```

Function listing:

```
    ∇ Z←X FITS Y;K;L
[1]   ⍝NUMBER OF CHARACTERS THAT FIT INTO SPACE WITHO
      UT DIVIDING WORDS
[2]   Z←¯1↑(K⍴~L)/⍳K←¯1↑(L←' '=(X+1)↑Y)/⍳X+1
[3]   ⍝840525 07.40
    ∇
```

Name: IOTΔ1 - X ⍳ Y for integers between ⎕IO,1999+⎕IO

Syntax: Z←X IOTΔ1 Y

Description:

X: Integer vector with elements between ⎕IO,1999+⎕IO
Y: Integer scalar or vector of the same domain
Z: Integer scalar or vector of length Y that returns those indices of X where the elements of Y are found first in X.

This function works like the primitive dyadic function ⍳ , but is more effective.

Example:

```
      30 20 20 40 IOTΔ1 ⎕←10×⍳5
10 20 30 40 50
5 2 1 4 5
```

Function listing:

```
    ∇ Z←X IOTΔ1 Y
[1]  ⍝X ⍳ Y FOR INTEGERS BETWEEN ⎕IO,1999+⎕IO
[2]   Z←2000⍴⎕IO+⌈/⍴X
[3]   Z[⍋X]←⍋⍳⍴X
[4]   Z←Z[Y]
[5]  ⍝840723 10.22
    ∇
```

Name: NC1 - Returns 1 if Y can be a single executable number

Syntax: Z←X NC1 Y

Description:

X: 0 or 1
Y: Character string (scalar or vector)
Z: 1 if Y can be a single executable number, else 0. If X is 1 the function returns 1 even though Y is empty.

Example:

```
      0 NC1 ' 1.2E¯12  '
1
      1 NC1 ' 1.2E¯12  '
1
      0 NC1 ' '
0
      1 NC1 ' '
1
```

Function listing:

```
    ∇ Z←X NC1 Y;E;I;L;N;P;S;⎕IO
[1]   ⍝⍝RETURNS 1 IF Y CAN BE A SINGLE EXECUTABLE NUM
      BER
[2]   ⍝IF X IS 1 Y MAY BE EMPTY
[3]   ⎕IO←1
[4]   Z←0
[5]   →E/0,Z←X^E←^/L←Y=' '
[6]   →0×⍳~Z←~' '∊Y←((∨\L)^⌽∨\⌽L←~L)/Y
[7]   →0×⍳Z←^/N←Y∊'0123456789'
[8]   S←Y∊'-¯'
[9]   P←Y='.'
[10]  I←¯1+(E←Y='E')⍳1
[11]  Z←(^/N∨S∨P∨E)^(~∨/1↓I↑S)^(1≥+/I↑P)^(∨/I↑N)
[12]  →((~Z)∨I=⍴Y)/0
[13]  Z←Z^~∨/(I←I+1)↓E
[14]  Z←Z^(~∨/1↓I↓S)^(~∨/I↓P)^(</ 2 0 <+/I↓N)
[15] ⍝840531 13.44
    ∇
```

Name: NUM - Test if array X is numeric

Syntax: Z←NUM X

Description:

X: Any array
Z: Scalar 1 if X is numeric, else 0

Example:

```
      NUM 5 6 3⍴⍳30
1
      NUM 'JOT'
0
```

Function listing:

```
    ∇ Z←NUM X
[1]  ⍝TEST IF ARRAY X IS NUMERIC
[2]   Z←0∊0\0⍴X
[3]  ⍝840801 08.49
    ∇
```

Name: RIOTΔ2 - Finds X ⍳ Y by rows for numeric arrays

Syntax: Z←X RIOTΔ2 Y

Description:

X: Numeric scalar, vector, or matrix
Y: Numeric array
Z: Integer array of shape ¯1↓⍴Y (1 for a scalar or vector Y) showing where in X the corresponding rows of Y are found first

Where a row of Y is not found in X,the corresponding element of the result is assigned a value that exceeds the highest existing row number in X by one. This function is based on ⍋ and =. The result is origin dependent. For character arrays there is another auxiliary function RIOTΔ1.

Example:

```
      M←⍉4 3⍴⍳12
      M
 1  4  7 10
 2  5  8 11
 3  6  9 12
      N←(3 4⍴1 2),[.1]⊖M
      N
 1  2  1  2
 1  2  1  2
 1  2  1  2

 3  6  9 12
 2  5  8 11
 1  4  7 10
      M RIOTΔ2 N
4 4 4
3 2 1
      M RIOTΔ2 3 6 9 12
3
```

Function listing:

```
     ∇ Z←X RIOT⍙2 Y;DM;DX;I;P;RXO
[1]   ⍝⍝FINDS X ⍳ Y BY ROWS FOR NUMERIC ARRAYS
[2]   ⍝BASED ON ⍋ AND =
[3]    DX←⍴Y←((×/¯1↓RXO),¯1↑1,RXO←⍴Y)⍴Y
[4]    DM←⍴X←(¯2↑ 1 1 ,⍴X)⍴X
[5]    P←1↓(DX⌊DM)+DX≠DM
[6]    Y←((1↑DX),P)↑Y
[7]    X←((1↑DM),P)↑X
[8]    I←⍳1↑DX+DM
[9]    P←×/P-~⍳1
[10] R1:I←I[⍋(X[;P],Y[;P])[I]]
[11]   →R1⌈⍳⎕IO≤P←P-1
[12]   X←X,[⍳1] Y
[13]   P←1↓∨/Y≠¯1⊖Y←X[I;]
[14]   Z←(¯1↓RXO)⍴(((1,P)/I)⌊⎕IO+1↑DM)[(1↑DM)↓(+\⎕IO,
       P)[⍋I]]
[15] ⍝840608 10.47
     ∇
```

Name: RNDM - x/X random numbers from Y[1] to Y[2] in Y[3] decimals

Syntax: Z←X RNDM Y

Description:

X: Non-negative integer scalar or vector
Y: Numeric scalar or vector of two or three elements
Z: Array of shape X returning x/X random numbers from Y[1] to Y[2] in Y[3] decimals. The default value for Y[3] is 0. If Y[3] is negative, the numbers are rounded to 1E¯Y[3].

If Y is a positive scalar, the default value for Y[2] is 2×Y. If Y is a negative scalar, the value is zero.

Example:

```
      3 6 RNDM 4 6 3
4.609 4.983 5.069 4.305 5.812 4.564
4.859 4.616 4.323 5.599 4.182 4.133
5.663 4.046 5.047 4.846 4.315 5.569
      6 RNDM 0 10000 ¯2
9900 6100 7300 9600 2200 3600
      10 RNDM 20
27 20 27 20 37 23 38 38 38 20
```

Function listing:

```
    ∇ Z←X RNDM Y
[1]  ⍝X/X RANDOM NUMBERS FROM Y[1] TO Y[2] IN Y[3] D
     ECIMALS
[2]  ⍝⍴Z ←→ X
[3]   Z←(Y[⍳1]-Z×⎕IO)+(Z←10*-¯1↑Y)×?X⍴1+(|-/2↑Y)×10*
     ¯1↑Y←3↑Y,0
[4]   ⊥(0=¯1↑Y)/'Z←⌊Z'
[5]  ⍝840801 10.09
    ∇
```

Name: RNDMN - X random numbers from normal distribution

Syntax: Z←X RNDMN Y

Description:

X: Non-negative integer
Y: Numeric vector
Z: Vector containing X random numbers from normal distribution with average Y[1] and variance Y[2]

Example:

```
      AVE←10
      VAR←1.2
      4 RNDMN AVE,VAR
10.75906802 9.789222419 8.466313745 10.17979441
      VAR←(+/(V-+/V÷⍴V)*2)÷⍴V←1000 RNDMN AVE,VAR
      VAR
1.143265543
      +/V÷⍴V
10.02093234
```

Function listing:

```
    ∇ Z←X RNDMN Y;C;⎕IO
[1]   ⍝⍝X RANDOM NUMBERS FROM NORMAL DISTRIBUTION
[2]   ⍝WITH AVERAGE Y[1] AND VARIANCE Y[2]
[3]   ⎕IO←1
[4]   Z←(C←2,C÷2)⍴0.000001×?(C←X+2|X)⍴1000000
[5]   Z←X⍴⍉Y[1]+(C⍴((¯2×Y[2])×⍟Z[1;])*0.5)× 2 1 ∘.○
      2×Z[2;]
[6]   ⍝840606 11.38
    ∇
```

Name: ROUND - Rounds Y to X decimals

Syntax: Z←X ROUND Y

Description:

X: Integer
Y: Numeric array
Z: Array Y with its elements rounded to X decimals.
A negative left argument rounds the elements of Y to 1E¯X.

Example:

```
      M←2 4⍴⍳8
      M
0          0.6931471806 1.098612289  1.386294361
1.609437912  1.791759469  1.945910149  2.079441542
      2 ROUND M
0    0.69 1.1  1.39
1.61 1.79 1.95 2.08
      N←1E6×M
      N
0        693147.1806 1098612.289  1386294.361
1609437.912  1791759.469  1945910.149  2079441.542
      ¯3 ROUND N
0  693000 1099000 1386000
1609000 1792000 1946000 2079000
```

Function listing:

```
    ▽ Z←X ROUND Y
[1]  ⍝ROUNDS Y TO X DECIMALS
[2]   →(X=0)/S
[3]   Z←(10*-X)×⌊0.5+Y×10*X
[4]   →0
[5]  S:Z←⌊0.5+Y
[6]  ⍝840801 08.59
    ▽
```

Name: TO - Arithmetic progression from X to Y

Syntax: Z←X TO Y

Description:

X: Integer
Y: Integer
Z: Arithmetic progression from X to Y - step 1 or ¯1

The function AP may prove useful if progressions of various steps are needed.

Example:

```
      7 TO 1
7 6 5 4 3 2 1
      2 TO 6.5
  11 DOMAIN ERROR
TO[2] Z←(X-⎕IO××Z)+(×Z)×⍳1+|Z←Y-X
                          ^
```

Function listing:

```
    ∇ Z←X TO Y
[1]  ⍝ARITHMETIC PROGRESSION FROM X TO Y
[2]   Z←(X-⎕IO××Z)+(×Z)×⍳1+|Z←Y-X
[3]  ⍝840801 09.01
    ∇
```

Name: UNIQ - Unique elements of X by upgrade method

Syntax: Z←UNIQ X

Description:

X: Numeric array
Z: Vector consisting of unique elements of X in increasing order.

Example:

```
      M←2 2 5⍴?⍵\14
      M
 8 11  9  7 10
 6  6  6  5  3

 2  3  1  1  8
11  9  7 10  6
      UNIQ M
1 2 3 5 6 7 8 9 10 11
```

Function listing:

```
    ∇ Z←UNIQ X
[1]  ⍝UNIQUE ELEMENTS OF X BY UPGRADE METHOD
[2]   →0×⍳0=⍴Z←X≠¯1⌽X←X[⍋X←,X]
[3]   Z←(1,1↓Z)/X
[4]  ⍝840801 09.02
    ∇
```

Name: UNIQL - Logical vector indicating the first positions of elements

Syntax: Z←UNIQL X

Description:

X: Numeric vector
Z: Logical vector that contains 1 for each first occurrance of an element

Example:

```
      V←3 3 2 4 3 7 2 2
      W←UNIQL V
      W
1 0 1 1 0 1 0 0
      W/V
3 2 4 7
```

Function listing:

```
    ▽ Z←UNIQL X;I
[1]  ⍝LOGICAL VECTOR INDICATING THE FIRST POSITIONS
      OF ELEMENTS
[2]   Z←(X/⍴X)↑1
[3]   →(1≥⍴Z)/0
[4]   X←X[I←⍋X]
[5]   Z←(0,(1↓X)≠¯1↓X)
[6]   Z[I]←~Z
[7]  ⍝840801 09.05
    ▽
```

Name: UNIQ2 - Unique elements of vector Y by index method

Syntax: Z←X UNIQ2 Y

Description:

X: Two-element numeric vector. 0<(X[2]-X[1])<400000
Y: Numeric vector, whose elements are within the limits X[1] ≤ Y ≤ X[2].
Z: Vector containing the unique elements of Y in increasing order

The number of elements in the left argument is unlimited but only the two first ones are significant. This function uses the index method, and it is more efficient than the corresponding auxiliary function UNIQ.

Example:

```
      0 10 UNIQ2 ¯1 10 ¯1 ¯1 2 3 2 2
   3 INDEX ERROR
UNIQ2[5] Z[Y-X[1]-1]←1
           ^
      ¯1 10 UNIQ2 ¯1 10 ¯1 ¯1 2 3 2 2
¯1 2 3 10
```

Function listing:

```
    ∇ Z←X UNIQ2 Y;⎕IO
[1]  ⍝⍝UNIQUE ELEMENTS OF VECTOR Y BY INDEX METHOD
[2]  ⍝X[1] ≤ Y ≤ X[2]  AND 0 < (X[2]-X[1]) < 400000
[3]   ⎕IO←1
[4]   Z←(1+X[2]-X[1])⍴0
[5]   Z[Y-X[1]-1]←1
[6]   Z←(X[1]-1)+Z/⍳⍴Z
[7]  ⍝840801 09.17
    ∇
```

Name: XNC - Executes character array Y containing X numbers

Syntax: Z←X XNC Y

Description:

X: Positive integer
Y: Character array
Z: Integer vector of length X. (,Y) is divided into X strings. These strings are executed if they are integers, and are returned in the corresponding elements of Z. Non-integer strings are returned in Z as zeroes.

This function produces another result in a global variable RC. RC is a logical vector of length X with those elements corresponding to non-integer values of Y set to 1. This function needs the auxiliary function DABA.

Example:

```
      M←'/' LIST 'MATRIX/¯2 300/E20399/34 5.0'
      M
MATRIX
¯2 300
E20399
34 5.0
      4 XNC M
0 ¯2300 0 0
      RC
1 0 1 1
      RC/M
MATRIX
E20399
34 5.0
      8 XNC M
0 0 ¯2 300 0 399 34 0
      RC
1 1 0 0 1 0 0 1
```

Function listing:

```
     ∇ Z←X XNC Y;N
[1]   ⍝⍝EXECUTES CHARACTER ARRAY Y CONTAINING X NUMBE
      RS
[2]   ⍝ZEROS NON-NUMERIC VALUES AND SETS CORRESPONDIN
      G RC VALUES TO 1
[3]   ⍝NEEDS DABA
[4]    Z←X⍴0
[5]    →0×⍳0∈⍴Y←DABA(X,⌊(×/⍴Y)÷X)⍴Y
[6]    RC←~Z←∧/Y∈'¯-0123456789 '
[7]    RC←RC∨~N←((N=1)≤((0 1 ↓Y)∨.≠' ')∧Y[;⎕IO]∈'-¯')
       ∧1≥N←+/Y∈'-¯'
[8]    Y←,' ',(Z←Z∧N∧Y∨.≠' ')≠Y
[9]    Y[(Y='-')/⍳⍴Y]←'¯'
[10]   Z←Z\1↓⊥'0 ',Y
[11]  ⍝840801 10.00
     ∇
```

Name: XNCD - Executes character array Y containing X numbers, also decimals

Syntax: Z←X XNCD Y

Description:

X: Positive integer
Y: Character array
Z: Numeric vector of length X. (,Y) is divided into X strings. These strings are executed if they are numeric (non-exponential), and are returned in the corresponding elements of Z. Non-numeric strings are set zero.

This function produces another result in a global variable RC, so that if there are any non-numeric strings in Y RC will be set to 1, else to 0.
This function needs the auxiliary function DABA.

Example:

```
      ⎕←M←'∘'LIST'MATRIX∘¯2 300∘1E2399∘34 5.0'
MATRIX
¯2 300
1E2399
34 5.0
      8 XNC M
0 0 ¯2 300 0 399 34 0
      8 XNCD M
0 0 ¯2 300 0 399 34 5
      RC
1
```

Function listing:

```
    ∇ Z←X XNCD Y;N
[1]   ⍝⍝EXECUTES CHARACTER ARRAY Y CONTAINING X NUMBE
      RS, ALSO DECIMALS
[2]   ⍝ZEROS NON-NUMERIC VALUES AND SETS RC←1
[3]   ⍝NEEDS DABA
[4]   RC←~∧/Z←∧/(Y←DABA(X,⌊(X/⍴Y)÷X)⍴Y)∈',¯-01234567
      89 '
[5]   RC←RC∨~∧/N←(1≥Y+.='.')∧((N=1)≤((0 1 ↓Y)∨.≠' ')
      ∧Y[;⎕IO]∈'-¯')∧1≥N←+/Y∈'-¯'
[6]   Y←,' ',(Z←Z∧N∧Y∨.≠' ')⌿Y
[7]   Y[(Y='-')/⍳⍴Y]←'¯'
[8]   Z←Z\1↓⊥'0 ',Y
[9]   ⍝840801 10.06
    ∇
```

Name: ZDIV - Devides X by Y giving 0 if Y=0

Syntax: Z←X ZDIV Y

Description:

X: Numeric scalar or an array of the shape of Y
Y: Numeric scalar or an array of the shape of X
Z: The ordinary division of X by Y. If (an element of) Y is 0 the (corresponding element of) result is 0.

Example:

```
      □←M←2 3⍴⍳6
1 2 3
4 5 6
      □←N←2 3⍴0,⍳5
5 4 3
2 1 5
      M ZDIV N
0.2 0.5 1
2   5   1.2
```

Function listing:

```
    ∇ Z←X ZDIV Y
[1]  ⍝DEVIDES X BY Y
[2]  ⍝GIVES 0 IF Y=0
[3]   Z←(~Z)×X÷Y+Z←Y=0
[4]  ⍝841119 15.23
    ∇
```

Name: BES - Catenates matrices along the last dimension

Syntax: Z←X BES Y

Description:

X: Scalar, vector or matrix
Y: Scalar, vector or matrix of type X
Z: Arrays X and Y catenated along the last dimension. If X and Y are not of correct shapes the smaller one is padded with 0's or blanks.

Example:

```
      M←3 6 ⍴⍳⍴M 4 6 2
      M
4.31 4.38 4.05 5.12 5    4.23
5.49 4.81 5.3  4.21 4.01 4.7
5.67 4.02 5.47 4.26 4.36 4.02
      (CL⍴M⍳6) BES M
1    4.31 4.38 4.05 5.12 5    4.23
2    5.49 4.81 5.3  4.21 4.01 4.7
3    5.67 4.02 5.47 4.26 4.36 4.02
4    0    0    0    0    0    0
5    0    0    0    0    0    0
6    0    0    0    0    0    0
      ((⍕CL⍴M⍳6),'|') BES ⍕ M
1|4.31 4.38 4.05 5.12 5    4.23
2|5.49 4.81 5.3  4.21 4.01 4.7
3|5.67 4.02 5.47 4.26 4.36 4.02
4|
5|
6|
```

Function listing:

```
    ∇ Z←X BES Y;C;RA;RB
[1]   ⍝⍝CATENATES MATRICES ALONG THE LAST DIMENSION
[2]   ⍝X AND Y CAN ALSO BE VECTORS
[3]    X←(RA←¯2↑ 1 1 ,⍴X)⍴X
[4]    Y←(RB←¯2↑ 1 1 ,⍴Y)⍴Y
[5]    C←1↑RA⌈RB
[6]    X←(RA⌈C,0)↑X
[7]    Y←(RB⌈C,0)↑Y
[8]    Z←X,Y
[9]   ⍝840801 12.47
    ∇
```

Name: BES1 - Catenates a vector and a matrix along the last dimension

Syntax: Z←X BES1 Y

Description:

X: Scalar, vector or matrix
Y: Scalar, vector or matrix of type X
Z: Catenates the arguments along the last dimension. A vector argument is multiplied to conform with a matrix argument.

If both arguments are matrices the left one is raveled prior to catenation.

Example:

```
      M←(⍕CLM⍳5),' :  ' BES1 ⍉5 6 RNDM 4 6 2
      M
1 :  5.05 5.09 5.43 5.54 5.23 5.73
2 :  5.21 4.33 5.66 5.4  5.36 4.18
3 :  5.14 4.8  4.15 4.67 5.7  4.82
4 :  4.71 4.99 4.06 5.15 5.86 5.39
5 :  4.74 4.62 5.53 4.89 5.7  5.69
      'EXAMPLE ' BES1 M
EXAMPLE 1 :  5.05 5.09 5.43 5.54 5.23 5.73
EXAMPLE 2 :  5.21 4.33 5.66 5.4  5.36 4.18
EXAMPLE 3 :  5.14 4.8  4.15 4.67 5.7  4.82
EXAMPLE 4 :  4.71 4.99 4.06 5.15 5.86 5.39
EXAMPLE 5 :  4.74 4.62 5.53 4.89 5.7  5.69
```

Function listing:

```
    ∇ Z←X BES1 Y
[1]  ⍝⍝CATENATES A VECTOR AND A MATRIX ALONG THE LAS
      T DIMENSION
[2]  ⍝MULTIPLIES THE VECTOR ARGUMENT TO CONFORM WITH
       THE MATRIX ARGUMENT
[3]   →(1≥⍴⍴Y)/L1
[4]   Z←(((1⍴⍴Y),⍴X)⍴X←,X),Y
[5]   →0
[6]  L1:Z←X,((1⍴⍴X),⍴Y)⍴Y←,Y
[7]  ⍝840801 12.56
    ∇
```

Name: CHNG - Replaces the elements of Y found in X[1;] to those in X[2;]

Syntax: Z←X CHNG Y

Description:

X: Two-row matrix
Y: Array of the same type (character or numeric)
Z: Array Y with the elements found in X[1;] replaced by the corresponding elements in X[2;]

Example:

```
      ⎕←N←10 2⍴¯500+?3 4⍴1000
 ¯127.00   ¯411.00    336.00   ¯150.00
 ¯166.00   ¯320.00    478.00    278.00
  284.00   ¯210.00    127.00   ¯386.00
      ⎕←M←2 2⍴'¯.-,'
¯.
-,
      M CHNG N
 -127,00   -411,00    336,00   -150,00
 -166,00   -320,00    478,00    278,00
  284,00   -210,00    127,00   -386,00
      N←3 3⍴1 0 0 0
      N
1 0 0
0 1 0
0 0 1
      (2 2⍴1 0 8 1) CHNG N
8 1 1
1 8 1
1 1 8
```

Function listing:

```
    ∇ Z←X CHNG Y;D;L;⎕IO;I
[1]   ⍝REPLACES THE ELEMENTS OF Y FOUND IN X[1;] TO T
      HOSE IN X[2;]
[2]   ⎕IO←1
[3]   D←⍴Y
[4]   I←X[1;]⍳Y←,Y
[5]   L←I≤1↓⍴X
[6]   Y[L/⍳⍴L]←(X[2;])[L/I]
[7]   Z←D⍴Y
[8]   ⍝840606 16.37
    ∇
```

Name: CHNG1 - Replaces X[1] by X[2] in an array Y

Syntax: Z←X CHNG1 Y

Description:

X: Two-element vector
Y: Array of type X
Z: Array Y with the elements equal to X[1] replaced by X[2]

Example:

```
      M←¯3+?3 3⍴⍳9
      M
¯2 ¯2 ¯2
¯1  2 ¯1
 3 ¯1  2
      '¯-' CHNG1 ⍉ M
-2 -2 -2
-1  2 -1
 3 -1  2
```

Function listing:

```
    ∇ Z←X CHNG1 Y
[1]  ⍝REPLACES X[1] BY X[2] IN AN ARRAY Y
[2]   Z[(Z=1⍴X)/⍳⍴Z←,Y]←1↓X
[3]   Z←(⍴Y)⍴Z
[4]  ⍝840607 10.03
    ∇
```

Name: DABA - Moves all blanks in an array X to the end of each line

Syntax: Z←DABA X

Description:

X: Character array
Z: Array X with all blanks moved to the end of each line

Example:

```
      M←'/'LISI'UN NECESSARY/ BLANKS / 3 , 88'
      M
UN NECESSARY
 BLANKS
 3 , 88
      DABA M
UNNECESSARY
BLANKS
3,88
```

Function listing:

```
    ∇ Z←DABA X;L
[1]  ⍝MOVES ALL BLANKS IN AN ARRAY X TO THE END OF E
     ACH LINE
[2]   Z←⍴X
[3]   X←(,L←X≠' ')/,X
[4]   X←(,(+/L)∘.>(-⎕IO)+⍳¯1↑Z)\X
[5]   Z←Z⍴X
[6]  ⍝840607 10.47
    ∇
```

Name: DMBA - Moves all multiple blanks in an array X to the end of each line

Syntax: Z←DMBA X

Description:

X: Character array
Z: Array X with all multiple and leading blanks moved to the end of each line.

Example:

```
      ⎕←M←2 3 20⍴'      WORDS   WORDS '
     WORDS   WORDS
    WORDS   WORDS
   WORDS   WORDS

  WORDS   WORDS
 WORDS   WORDS
WORDS   WORDS      W
      ⎕←M←DMBA M
WORDS WORDS
WORDS WORDS
WORDS WORDS

WORDS WORDS
WORDS WORDS
WORDS WORDS W
      ⍴M
2 3 20
```

Function listing:

```
    ∇ Z←DMBA X;L
[1]  ⍝MOVES ALL MULTIPLE BLANKS IN AN ARRAY X TO THE
       END OF EACH LINE
[2]   ⍎(0∈⍴X)/'→X/⍴Z←X'
[3]   L←((Z←(-⍴⍴L)↑1)↓L,0)∨L←' '≠X←' ',X
[4]   Z←Z↓(⍴X)⍴(,(+/L)∘.>(-⎕IO)+⍳¯1↑⍴X)\(,L)/,X
[5]  ⍝840801 11.20
    ∇
```

Name: DTB⍙ - Drops trailing blanks from an array X

Syntax: Z←DTB⍙ X

Description:

X: Character array
Z: Array X with trailing blanks dropped from the last axis

Example:

```
      M←'/'LIST'WHY SHOULD/   WE/WASTE SPACE
      ⍴⎕←M
WHY SHOULD
   WE
WASTE SPACE
3 17
      ⎕←M←DTB⍙ M
WHY SHOULD
   WE
WASTE SPACE
      ⍴M
3 11
```

Function listing:

```
    ∇ Z←DTB⍙ X
[1]  ⍝DROPS TRAILING BLANKS FROM AN ARRAY X
[2]   Z←((-⍴⍴X)↑⌈/(1⌈⍴Z)↑Z←,1-(X=' ')⊥1)↓X
[3]  ⍝840619 12.32
    ∇
```

Name: EQ - Compares character matrix X to vector Y by rows

Syntax: Z←X EQ Y

Description:

X: Character matrix (any array)
Y: Character scalar or vector
Z: Logical vector of as many elements as there are rows in X. The elements corresponding to the rows of X that equal Y are set to 1. Y and a row R are considered equal if R∧.=((⍴Y)⌊1↓⍴X)↑Y. Leading blanks of Y are dropped before the comparisons.

X may be a non-matrix array, too. In that case the function works like the dyadic primitive function ∈.

Example:

```
      ⎕←M←'∘'LIST'ABC∘ ABC∘EFGH∘ABC'
ABC
 ABC
EFGH
ABC
      M EQ 'ABC'
1 0 0 1
      M EQ ' ABC'
1 0 0 1
      M EQ 'A'
1 0 0 1
      M EQ 'EFGHIJKL'
0 0 1 0
```

Function listing:

```
    ∇ Z←X EQ Y;D;I;N;P
[1]   ⍝COMPARES CHARACTER MATRIX X TO VECTOR Y BY ROW
      S
[2]   →LB⌈⌿2=⍴⍴X
[3]   Z←1=X∈Y
[4]   →0
[5]  LB:Z←(D←1⍴⍴X)⍴0
[6]   Y←(∨⌿Y≠' ')/Y
[7]   I←(X[;⎕IO]=1↑Y)/⍳D
[8]   I←(X[I;⍳P]∧.=Y[⍳P←(⍴Y)⌊1↓⍴X])/I
[9]   Z[I]←1
[10] ⍝840802 07.46
    ∇
```

Name: FLEFT - Rotates X so that it prints flushleft

Syntax: Z←FLEFT X

Description:

X: Character array
Z: Array X rotated along the last axis so that it prints flushleft. ⍴Z ↔ ⍴X.

Example:

```
      M←4 14⍴'JANET + JOY    '
      M
JANET + JOY
 JANET + JOY
  JANET + JOY
   JANET + JOY
      FLEFT M
JANET + JOY
JANET + JOY
JANET + JOY
JANET + JOY
```

Function listing:

```
    ∇ Z←FLEFT X
[1]  ⍝ROTATES X SO THAT IT PRINTS FLUSHLEFT
[2]   Z←(+/∧\X=' ')⌽X
[3]  ⍝840607 11.39
    ∇
```

Name: FRIGHT - Rotates X so that it prints flushright

Syntax: Z←FRIGHT X

Description:

X: Character array
Z: Array X rotated along the last axis so that it prints flushright. ⍴Z ↔ ⍴X.

Example:

```
      M←4 14⍴'BILL + GUY      '
      M
BILL + GUY
  BILL + GUY
    BILL + GUY
      BILL + G
      FRIGHT M
    BILL + GUY
    BILL + GUY
    BILL + GUY
      BILL + G
```

Function listing:

```
     ∇ Z←FRIGHT X;⎕IO
[1]   ⍝ROTATES X SO THAT IT PRINTS FLUSHRIGHT
[2]    ⎕IO←1
[3]    Z←(((X≠' ')⌈.×⍳Z)-Z←¯1↑⍴X)⌽X
[4]   ⍝840607 12.01
     ∇
```

Name: IN - Returns X and gives a global variable M a value Y

Syntax: Z←X IN Y

Description:

X: Any array
Y: Any array
Z: Argument X

Global variable M is given a value Y.
This function is designed to make the use of the auxiliary functions REPLACES, REPLACESBR, and EPUT more convenient.

Example:

```
      ⎕←V←'JKLMN'REPLACES'JKLMN'IN'ABCDEFGHIJKLMNOP'
ABCDEFGHIJKLMNOP
      (2 8⍴V)EPUT'▽'IN ⎕←2 3⍴2 2 1 2 6 6
2 2 1
2 6 6
ABCDE▽GH
I▽KLM▽OP
```

Function listing:

```
    ▽ Z←X IN Y
[1]  ⍝RETURNS X AND GIVES A GLOBAL VARIABLE M A VALU
      E Y
[2]   M←Y
[3]   Z←X
[4]  ⍝840801 11.34
    ▽
```

Name: REPLACES - String X replaces string Y in global variable M

Syntax: Z←X REPLACES Y

Description:

X: Scalar or vector
Y: Scalar or vector of type X (character or numeric)
Z: Global variable M with strings Y replaced by a string X. A string as a whole has to be in a same row in ordeF to become replaced.

Global variable M can be any array of type X. The function expunges M. This function is used together with the auxiliary function IN.

Example:

```
      NAMES←' 'LIST'MARY MICHEL FLOR ANGIE'
      NAMES
MARY
MICHEL
FLOR
ANGIE
      'ELINE' REPLACES 'EL' IN NAMES
MARY
MICHELINE
FLOR
ANGIE
```

Function listing:

```
    ▽ Z←X REPLACES Y;D;EX;I;J;L;M;P;PJ;PN;PO;T;⍙P;⎕I
      O
[1]   ⍝STRING X REPLACES STRING Y IN GLOBAL VARIABLE
      M
[2]   ⎕IO←1
[3]   D←(×/EX←¯1↓⍴Z),¯1↑1,⍴Z←M
[4]   J←⎕EX 'M'
[5]   J←((1+D[2]|J-1)≤1+D[2]-PO←⍴Y)/J←(M=1↑Y←,Y)/⍳⍴M
      ←,Z
[6]   →0×⍳0=PJ←⍴J←(M[J∘.+¯1+⍳PO]∧.=Y)/J
[7]   L←((PJ×⍙P←(PN←⍴,X)-PO)+×/D)⍴1
[8]   L[I←J+⍙P×¯1+⍳PJ]←1-⍙P
[9]   M←M[1⌈+\L]
[10]  M[,I∘.+¯1+⍳PN]←(PN×PJ)⍴X
[11]  P←D[2]+⍙P×+/(⍳D[1])∘.=⌈J÷D[2]
[12]  Z←(EX,¯1↑⍴T)⍴(,T←(((⍴P),L/P)⍴1),(P-L/P)∘.≥⍳(⌈/
      P)-L/P)\M
[13] ⍝840731 09.43
    ▽
```

Name: REPLACESBR - Rows of X replace rows of Y in global variable M

Syntax: Z←X REPLACESBR Y

Description:

X: Character matrix
Y: Character matrix containing rows of M.
   There must be equal number of rows in X and Y.
Z: Global variable M with rows given in Y
   replaced by corresponding rows in X. Array M is
   padded with blanks to conform with a row longer
   than the length of M's last axis.

Global variable M may be any character array. It is expunged during the execution of the function. The function is commonly used together with the auxiliary function IN. This function needs the auxiliary functions CHANGE, CHNG1, and LIST.

Example:

```
      □←N←2 3 6⍴' 'LIST'ABC DEFGHI JK LMNOP'
ABC
DEFGHI
JK

LMNOP
ABC
DEFGHI
      □←N←(2 20⍴'∘') REPLACESBR (2 3⍴'JK ABC') IN N
∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘
DEFGHI
∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘

LMNOP
∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘∘
DEFGHI
      ⍴N
2 3 24
```

Function listing:

```
     ∇ Z←X REPLACESBR Y;D;DO;I;J;K;M;PO;⎕IO
[1]   ⍝⍝ROWS OF X REPLACE ROWS OF Y IN GLOBAL VARIABL
       E M
[2]   ⍝X AND Y MUST BE MATRICES
[3]   ⍝NEEDS CHANGE CHNG1 LIST
[4]    ⎕IO←1
[5]    DO←⍴M
[6]    I←(Z∈Y[;1])/⍳⍴Z←,⎕AV[1],M
[7]    K←⎕EX 'M'
[8]    J←(Z,K⍴' ')[I∘.+¯1+⍳K←¯1↑D←⍴Y]+.=⍵' ⍝' CHNG1 Y
[9]    J←J=(⍴J)⍴PO←Y+.≠' '
[10]   J←J[K←(∨/J)/⍳⍴I;]⌈.×⍳1⍴D
[11]   Z←(Z,(K≠' ')/K←,X[J;]) CHANGE I[K],PO[J],(X+.≠
       ' ')[J]
[12]   Z←((¯1↓DO),¯1↑⍴Z)⍴Z←⎕AV[1] LIST Z
[13]  ⍝840801 12.26
     ∇
```

Name: REPS - Array ε by rows

Syntax: Z←X REPS Y

Description:

X: Array
Y: Array
Z: Array ε by rows. Array Z is of shape ¯1↓⍴X.
Trailing blanks or 0's are ignored.

Example:

```
      A←3 3 3⍴'ABC DEF '
      M←5 4⍴'F A DEF  EF    '
      A
ABC
 DE
F A

BC
DEF
 AB

C D
EF
ABC
      M
F A
DEF
 EF
  F
A DE
      A REPS M
0 0 1
0 1 0
0 0 0
      M REPS A
1 1 0 0 0
```

Function listing:

```
    ∇ Z←X REPS Y;RA;RB;R
[1]   ⍝ARRAY ε BY ROWS
[2]   ⍝(⍴Z) ≡ ¯1↓⍴X
[3]   X←(RA←(-1⌈⍴Z)↑1,Z←⍴X)⍴X
[4]   Y←(RB←(×/¯1↓RB),¯1↑1,RB←⍴Y)⍴Y
[5]   Z←(¯1↓Z)⍴∨/(((¯1↓RA),R)↑X)∧.=⍳(RB[⍳1],R←(¯1↑RA
      )⌈¯1↑⍴Y)↑Y
[6]   ⍝840802 08:19
    ∇
```

Name: RIOTΔ1 - Finds X ⍳ Y by rows for character arrays

Syntax: Z←X RIOTΔ1 Y

Description:

X: Character scalar, vector, or matrix
Y: Character array
Z: Integer array of shape ¯1↓⍴Y (1 for a vector or scalar Y) consisting of row numbers showing where in X the rows of Y are found first. Trailing blanks of the rows of X and Y are neglected.

Where a row of Y is not found in X the corresponding element of the result is assigned a value which exceeds the highest existing row number in X by one. This function is based on ⍋ and =. The result is origin dependent. For numeric arrays there is another auxiliary function RIOTΔ2.

Example:

```
      ITEMS←' 'LISΔ'TERMINAL PRINTER ASPIRIN'
      ITEMS
TERMINAL
PRINTER
ASPIRIN
      ITEMS RIOTΔ1 'PRINTER'
2
      ITEMS1←ITEMS,[1.5]' 'LISΔ'KEYBOARD PENCIL PHONE'
      ITEMS1
TERMINAL
KEYBOARD

PRINTER
PENCIL

ASPIRIN
PHONE
      ITEMS RIOTΔ1 ITEMS1
1 4
2 4
3 4
```

Function listing:

```
     ∇ Z←X RIOTA1 Y;A;DM;DX;I;J;P;RX0
[1]   ⍝⍝FINDS X ⍳ Y BY ROWS FOR CHARACTER ARRAYS
[2]   ⍝BASED ON ⍋ AND =
[3]    DX←⍴Y←((×/¯1↓RX0),¯1↑1,RX0←⍴Y)⍴Y
[4]    DM←⍴X←(¯2↑ 1 1 ,⍴X)⍴X
[5]    A←⍳P←1↓(DX⌊DM)+DX≠DM
[6]    Y←((1↑DX),P)↑Y
[7]    X←((1↑DM),P)↑X
[8]    I←⍳1↑DX+DM
[9]    Z←7
[10] R1:I←I[⍋(256⊥⎕AV⍳⍉X[;J],[⍳1] Y[;J←(-Z⌊⍴A)↑A])[I
       ]]
[11]   →R1⍴⍳0<⍴A←(-Z)↓A
[12]   X←X,[⍳1] Y
[13]   P←1↓∨/Y≠¯1⊖Y←X[I;]
[14]   Z←(¯1↓RX0)⍴(((1,P)/I)⌊⎕IO+1↑DM)[(1↑DM)↓(+\⎕IO,
       P)[⍋I]]
[15] ⍝840802 08.20
     ∇
```

Name: UNIQR - Drops multiple rows

Syntax: Z←UNIQR X

Description:

X: Character scalar, vector, or matrix
Z: Array X with multiple rows dropped

To minimize workspace consumption, this function uses the auxiliary function RIOTA1 instead of the inner product.

Example:

```
      ⎕←M←8 3⍴'VAWVOWOUI'
VAW
VOW
OUI
VAW
VOW
OUI
VAW
VOW
      UNIQR M
VAW
VOW
OUI
```

Function listing:

```
    ∇ Z←UNIQR X
[1]   ⍝⍝DROPS MULTIPLE ROWS
[2]   ⍝NEEDS RIOTA1
[3]    Z←X←(¯2↑ 1 1 ,⍴X)⍴X
[4]    Z←((⍳1↑⍴X)=X RIOTA1 X)⌿X
[5]   ⍝840801 12.37
    ∇
```

Name: UPON - Catenates matrices along the first dimension

Syntax: Z←X UPON Y

Description:

X: Scalar, vector or matrix
Y: Scalar, vector or matrix of type X
Z: Argument matrices catenated along the first dimension. Scalars and vectors are treated as 1-element or 1-row matrices. The smaller matrix is padded with 0's or blanks to conform with the larger one.

Example:

```
      M←3 4 RNDM 1 10 2
      M
2.36 4.28 7.13 5.11
8.47 7.77 7.04 6.2
1.45 8.54 7.88 2.93
      N←4 ROUND 4 2⍴*⍳8
      N
   2.7183    7.3891
  20.0855   54.5982
 148.4132  403.4288
1096.6332 2980.958
      N UPON M
   2.7183    7.3891    0         0
  20.0855   54.5982    0         0
 148.4132  403.4288    0         0
1096.6332 2980.958     0         0
   2.36      4.28      7.13      5.11
   8.47      7.77      7.04      6.2
   1.45      8.54      7.88      2.93
      'RANDOM' UPON 'NUMBERS:' UPON '-',[1] ⍕M
RANDOM
NUMBERS:
-----------------------
2.36 4.28 7.13 5.11
8.47 7.77 7.04 6.2
1.45 8.54 7.88 2.93
```

Function listing:

```
    ∇ Z←X UPON Y;DX;DY;W;⎕IO
[1]  ⍝CATENATES MATRICES ALONG THE FIRST DIMENSION
[2]   ⎕IO←1
[3]   W←⌈/1,(¯1↑DX←⍴X),¯1↑DY←⍴Y
[4]   Z←(((¯1↓DX),W)↑X),[0.5×⌈/1,(⍴DX),⍴DY]((¯1↓DY),
      W)↑Y
[5]  ⍝841119 15.57
    ∇
```

Name: VIOTA - Indices of the rows of Y in vector X

Syntax: Z←X VIOTA Y

Description:

X: Character scalar or vector (any array)
Y: Character array (not a scalar)
Z: Integer array of shape ¯1↓⍴Y (1 for a vector Y) consisting of indices which indicate where in X the rows of Y are found first

Where a row of Y is not found in X the corresponding element of the result is assigned a value which exceeds the length of X by one. Thus this function could be regarded as dyadic vector ⍳. All character arrays are accepted as the left argument, too, but they are raveled before the execution. The result is origin dependent. This function uses the auxiliary function CHNG1.

Example:

```
      ⎕←M←5 6⍴'12AB '
12AB 1
2AB 12
AB 12A
B 12AB
 12AB
      'AB 12AB 'VIOTA M
9 9 1 2 3
```

Function listing:

```
    ∇ Z←X VIOTA Y;W;R;D;I;P
[1]   ⍝⍝INDICES OF THE ROWS OF Y IN VECTOR X
[2]   ⍝VECTOR ⍳
[3]   ⍝NEEDS CHNG1
[4]    Z←(R←¯1↓D←⍴Y)⍴⎕IO
[5]    →(0=P←⍴X←,X)/0
[6]    I←(X∈(R,1)↑Y)/⍳P
[7]    Z←(' ⍒' CHNG1 Y)+.=(X,W⍴' ')[((⍳W←¯1↑D)-⎕IO)∘.
       +I]
[8]    Z←((⍳Z=(¯1⌽⍳⍴D)⍀((⍴I),R)⍴Y+.≠' ')⌈.×I+1
[9]    Z←(Z-~W)+(P+⎕IO)×W←Z=0
[10]  ⍝840802 08.37
    ∇
```

Name: CHANGE - Replaces substrings

Syntax: Z←X CHANGE Y

Description:

X: Vector containing the vector to be changed and the new substrings
Y: Integer vector of three or multiple of three elements: indices, old lengths and new lengths. The indices given in Y are the starting locations of the substrings to be replaced. The new lengths indicate the number of characters that are taken from the end of X to replace the characters whose number is given in the old lengths.
Z: Argument vector X with substrings indicated by Y replaced by substrings in the end of X

This function can also be used to replace parts of a numeric vector. The function calls the auxiliary function PIOTA.

Example:

```
      V←'VECTOR TO BE CHANGED'
      ⎕←V←(V,'A ','WAS') CHANGE 1 8 0 5 2 3
A VECTOR WAS CHANGED
```

Function listing:

```
    ∇ Z←X CHANGE Y;A;ΔP;L;I
[1]   ⍝⍝REPLACES SUBSTRINGS
[2]   ⍝X ↔ VECTOR, NEW SUBSTRINGS
[3]   ⍝Y ↔ INDICES, OLD LENGTHS, NEW LENGTHS
[4]   ⍝INDICES ARE THE LOCATIONS WHERE THE SUBSTRINGS
        TO BE REPLACED START
[5]   ⍝NEEDS PIOTA
[6]    Y←(3,(⍴Y)÷3)⍴Y
[7]    Z←(A←-+/Y[2+⎕IO;])↓X
[8]    L←((⍴Z)++/ΔP←-≠Y[2 1 +⎕IO;])⍴1
[9]    L[I←Y[⎕IO;]+0,¯1↓+\ΔP]←1-ΔP
[10]   Z←Z[⎕IO⌈(+\L)-~⎕IO]
[11]   Z[(I-⎕IO) PIOTA Y[2+⎕IO;]]←A↑X
[12] ⍝840802 10.59
    ∇
```

Name: DAB - Drops all blanks

Syntax: Z←DAB X

Description:

X: Character scalar or vector
Z: Vector X with all blanks deleted

Example:

```
      DAB 'A B C D E F '
ABCDEF
```

Function listing:

```
    ∇ Z←DAB X
[1]   ⍝DROPS ALL BLANKS
[2]   Z←(X≠' ')/X
    ∇
```

Name: DLB - Drops leading blanks

Syntax: Z←DLB X

Description:

X: Character scalar or vector
Z: Vector X with leading blanks deleted

Example:

```
      DLB '  A  B  C  D  E  '
A  B  C  D  E
```

Function listing:

```
    ∇ Z←DLB X
[1]   ⍝DROPS LEADING BLANKS
[2]   Z←(((,X≠' ')\1)-⎕IO)↓X
    ∇
```

Name: DMB - Drops multiple blanks

Syntax: Z←DMB X

Description:

X: Character scalar or vector
Z: Vector X with multiple blanks deleted

Example:

```
      DMB'    ABC     DEF     GHI  '
ABC DEF GHI
```

Function listing:

```
    ∇ Z←DMB X;L
[1]   ⍝DROPS MULTIPLE BLANKS
[2]   Z←(~1↑L)↓((1↓L,0)∨L←X≠' ')/X
    ∇
```

Name: DTB - Drops trailing blanks

Syntax: Z←DTB X

Description:

X: Character scalar or vector
Z: Vector X with trailing blanks deleted

Example:

```
      V←'  A  B  C  D  E        '
      ⍴V
24
      ⎕←V←DTB V
  A  B  C  D  E
      ⍴V
15
```

Function listing:

```
    ∇ Z←DTB X;⎕IO
[1]  ⍝DROPS TRAILING BLANKS
[2]   ⎕IO←1
[3]   Z←(¯1↑(X≠' ')/⍳⍴X)↑X
[4]  ⍝840619 12.32
    ∇
```

Name: INVECTOR - Gives the indices in vector Y where substring X is found

Syntax: Z←X INVECTOR Y

Description:

X: Scalar or vector (any array)
Y: Scalar or vector of type X
Z: Vector giving the indices in the vector Y where the substring X is found

The argument Y can be any array, but it is raveled to a vector prior to execution.

Example:

```
      'IN' INVECTOR 'INDICES FOUND IN A VECTOR'
1 15
```

Function listing:

```
    ∇ Z←X INVECTOR Y;J;RA;RB
[1]   ⍝GIVES THE INDICES IN VECTOR Y WHERE SUBSTRING
        X IS FOUND
[2]   J←(J≤⎕IO+RB-RA←⍴,X)/J←(Y=1↑X)/⍳RB←⍴Y←,Y
[3]   Z←(Y[J∘.+(⍳RA)-⎕IO]^.=X)/J
[4]   ⍝840618 15.36
    ∇
```

Name: NSS - Next substring of character vector named Y

Syntax: Z←X NSS Y

Description:

X: Character scalar
Y: Character vector
Z: The first substring of a character vector named Y. X acts as a separating character.

Substring Z is dropped from the vector ⊥Y.

Example:

```
      V←'1.WORD,2.WORD,,4.WORD'
      ',' NSS 'V'
1.WORD
      V
2.WORD,,4.WORD
      ',' NSS 'V'
2.WORD
      ',' NSS 'V'

      V
4.WORD
```

Function listing:

```
    ∇ Z←X NSS Y;E;S
[1]  ⍝NEXT SUBSTRING OF CHARACTER VECTOR NAMED Y
[2]  ⍝SUBSTRINGS ARE SEPARATED BY X
[3]  ⍝SUBSTRING IS DROPPED FROM THE VECTOR ⊥Y
[4]   E←⌊/(S←⊥Y)⍳X
[5]   Z←(E-⎕IO)↑S
[6]   ⊥Y,'←(E+~⎕IO)↓S'
[7]  ⍝840802 12.35
    ∇
```

Name: OE - X'th substring of character vector Y

Syntax: Z←X OE Y

Description:

X: Non-negative integer
Y: Vector
Z: X'th substring of vector Y. Substrings are separated by the first element of Y.

Example:

```
      3 OE ' TIMO SEPPO ARTO REIJO'
ARTO
```

Function listing:

```
     ∇ Z←X OE Y
[1]   ⍝⍝X'TH SUBSTRING OF CHARACTER VECTOR Y
[2]   ⍝1↑Y MUST CONTAIN THE SEPARATOR CHARACTER
[3]    Z←1↓(X=+\Y=1↑Y)/Y
[4]   ⍝840802 11.05
     ∇
```

Name: BLDMAT - Takes character input row by row and forms a matrix

Syntax: Z←BLDMAT

Description:

Z: Character matrix formed with row by row input. Each row is padded with blanks to conform with the longest row.

A carriage return indicates the end of input.

Example:

```
      M←BLDMAT
THESE ROWS
   ARE
INPUT FROM TERMINAL
      M
THESE ROWS
   ARE
INPUT FROM TERMINAL
```

Function listing:

```
    ▽ Z←BLDMAT;I;P;D;⎕IO
[1]  ⍝⍝TAKES CHARACTER INPUT ROW BY ROW AND FORMS A
      MATRIX
[2]  ⍝TO EXIT ENTER CR
[3]   D←⍴Z← 0 0 ⍴⎕IO←1
[4]  R1:→(R1+1)×⍳P←×/⍴I←⎕
[5]   ⊥(P<D[2])/'I←D[2]↑I'
[6]   ⊥(P>D[2])/'Z←(D[1],P)↑Z'
[7]   →R1,D←⍴Z←Z,[1] I
[8]  ⍝840801 12.59
    ▽
```

Name: CLM - Reshapes vector X into a one column matrix

Syntax: Z←CLM X

Description:

X: Character or numeric vector
Z: Vector X reshaped into a one column matrix

If argument X is not a vector this function returns it unchanged.

Example:

```
      CLM'APL AF'
A
P
L

A
F
      ⍉ CLM ⍳5
1
2
3
4
5
```

Function listing:

```
    ▽ Z←CLM X
[1]  ⍝RESHAPES VECTOR X INTO A ONE COLUMN MATRIX
[2]   →(1≠⍴⍴Z←X)/0
[3]   Z←((⍴X),1)⍴X
[4]  ⍝840801 08.14
    ▽
```

Name: DISPLAY - Displays character array Y in pages and columns

Syntax: Z←X DISPLAY Y

Description:

X: Non-negative integer vector or scalar. X[1] or scalar X indicates the number of 1↓⍴X arrays per column, X[2] the space between adjacent columns, and X[3] the number of adjacent columns per page. The default value for X[2] is 3, and for X[3] the maximum value set by ⎕PW.
Y: Character array of rank ≥2.
Z: Character array (of rank 1+⍴⍴Y) which displays Y according to the instructions given in X.

Example:

```
      ⎕←M←13 11⍴'1234567890'
12345678901
23456789012
34567890123
45678901234
56789012345
67890123456
78901234567
89012345678
90123456789
01234567890
12345678901
23456789012
34567890123
      2 5 3 DISPLAY M
12345678901     34567890123     56789012345
23456789012     45678901234     67890123456

78901234567     90123456789     12345678901
89012345678     01234567890     23456789012

34567890123

      ⎕PW←51
      A←9 2 2 6⍴M
      2 DISPLAY A
123456   901234   789012   234567   012345
789012   567890   345678   890123   678901

345678   123456   901234   456789   234567
901234   789012   567890   012345   890123


567890   345678   123456   678901
123456   901234   789012   234567

789012   567890   345678   890123
345678   123456   901231   456789
```

Function listing:

```
     ∇ Z←X DISPLAY Y;A;D;G;S;⎕IO
[1]   ⍝⍝DISPLAYS CHARACTER ARRAY Y IN PAGES AND COLUM
       NS
[2]   ⍝X[1] ↔ NUMBER OF (1↓⍴Y) ARRAYS PER COLUMN
[3]   ⍝X[2] ↔ SPACE BETWEEN ADJACENT COLUMNS (DEF. 3
       )
[4]   ⍝X[3] ↔ NUMBER OF ADJACENT COLUMNS PER PAGE (D
       EF. MAX SET BY ⎕PW)
[5]    ⎕IO←1
[6]    G←¯1↑D←⍴Y
[7]    S←1↑1↓(X←,X),2⍴3
[8]    A←⌈D[1]÷×/X←X[1],1↑(2↓X),L(⎕PW-2)÷G+S
[9]    Z←(A,(⍬X),1↓D)⍴((A××/X),1↓D)↑Y
[10]   Z←(1,(¯1⍬1+\⍴D),2+⍴D)⍉Z
[11]   Z←(A,X[1],(1↓¯1↓D),X[2]×G)⍴Z
[12]   Z←((X[2]×G+S)⍴(G⍴1),S⍴0)\Z
[13] ⍝840802 12.48
     ∇
```

Name: LIST - Lists Y using X as separator elements

Syntax: Z←X LIST Y

Description:

X: Scalar or vector
Y: Array of the same type as X
Z: Array Y reshaped to a matrix. The elements of Y between consequent separators indicated by X become an individual row in Z. The change of line means also separation. Rows are padded with 0's or blanks to conform with the longest row.

This function can be used to list numeric arrays,too, but it is most commonly found in connection with handling character information.

Example:

```
      M←'LET US MAKE'
      M←M,[.1]'A LIST. OK?'
      M
LET US MAKE
A LIST. OK?
      ' .' LIST M
LET
US
MAKE
A
LIST
OK?
      '/' LIST 'JOY/ANGIE/MICHEL'
JOY
ANGIE
MICHEL
```

Function listing:

```
    ∇ Z←X LIST Y;I;L;P
[1]  ⍝LISTS Y USING X AS SEPARATOR CHARACTERS
[2]   ⎕IO←1
[3]   L←(Y←(1↑X),,Y,''⍴X)∈X
[4]   Z←P∘.≥⍳0⌈⌈/P←¯1+(P>1)/P←(1↓I)-¯1↓I←L/⍳⍴Y
[5]   Z←(⍴Z)⍴(,Z)\(~L)/Y
[6]  ⍝840801 13.13
    ∇
```

Name: LISTO - Lists Y so that extra separators X produce empty lines

Syntax: M←X LISTO Y

Description:

X: Scalar or vector
Y: Array of type X
Z: List of Y formed in the way of the auxiliary function LISTI. The difference is that now each extra separator produces an empty line (row of 0's in the numeric case).

Example:

```
      '*' LISTO 'ROW 1***ROW 2*ROW 3**ROW 4'
ROW 1


ROW 2
ROW 3

ROW 4
```

Function listing:

```
     ∇ M←X LISTO Y;I;L;P;X;⎕IO
[1]   ⍝LISTS Y SO THAT EXTRA SEPARATORS X PRODUCE EMP
      TY LINES
[2]   ⎕IO←1
[3]   L←(Y←(1↑X),,Y,''⍴X)∈X
[4]   M←P∘.≥⍳0⌈⌈/P←¯1+(1↓I)-¯1↓I←L/⍳⍴Y
[5]   M←(⍴M)⍴(,M)\(~L)/Y
[6]   ⍝840801 13.21
     ∇
```

Name: MAT - Divides vector Y into a matrix according to lengths X

Syntax: Z←X MAT Y

Description:

X: Non-negative integer scalar or vector. +/X ↔ ⍴Y.
Y: Vector
Z: Vector Y divided into a matrix according to lengths X. All rows are padded with blanks or 0's to conform with the longest row.

Example:

```
      (⍳6) MAT 21⍴100
100   0   0   0   0   0
100 100   0   0   0   0
100 100 100   0   0   0
100 100 100 100   0   0
100 100 100 100 100   0
100 100 100 100 100 100
      ABC←'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
      (|¯6+⍳11) MAT 30⍴ABC
ABCDE
FGHI
JKL
MN
O

P
QR
STU
VWXY
ZABCD
```

Function listing:

```
    ∇ Z←X MAT Y;⎕IO
[1]  ⍝DIVIDES VECTOR Y INTO A MATRIX ACCORDING TO LE
      NGTHS X
[2]   ⎕IO←1
[3]   Z←(⍴Z)⍴(,Z←X∘.≥⍳⌈/0,X)\Y
[4]  ⍝840801 13.25
    ∇
```

Name: MATRIX - Reshapes X to a matrix

Syntax: Z←MATRIX X

Description:

X: Any array
Z: Array X reshaped to a matrix.

Example:

```
      M←MATRIX 9
      ⍴M
1 1
      M←3 2 8⍴ABC
      M
 ABCDEFG
HIJKLMNO

PQRSTUVW
XYZABCDE

FGHIJKLM
NOPQRSTU
      M←MATRIX M
      M
 ABCDEFG
HIJKLMNO
PQRSTUVW
XYZABCDE
FGHIJKLM
NOPQRSTU
```

Function listing:

```
    ∇ Z←MATRIX X
[1]  ⍝RESHAPES X TO A MATRIX
[2]   Z←((×/¯1↓⍴X),¯1↑1,⍴X)⍴X
[3]  ⍝840801 13.29
    ∇
```

Name: TABULATE - Tabulates data in matrix Y according to the Keys in X

Syntax: Z←X TABULATE Y

Description:

X: Character vector with Key variables separated by a comma

Y: Numeric matrix of data with columns corresponding to Key variables besides the columns containing the data to be tabulated. If one of the Key variables is of character type, Y is a character vector with Key data variables and the data variable separated from each other by a comma.

Z: Array containing one dimension for each Key variable and an additional dimension of length N←(¯1↑⍴Y) minus the number of Key variables in X if N>1. The data part of Y is tabulated according to the Keys given in X.

This function needs the auxiliary functions RIOTA1, NSS, and SUBSUM.

Example:

```
      DEPT←10 20 30 ◊ DEPTV←10 10 20 20 30
      SEX←1 2 ◊ SEXV←1 2 1 2 1
      ⎕←M←DEPTV,SEXV,5 4⍴45 23 12 23 90 12 34 44 21
10  1 45 23 12 23
10  2 90 12 34 44
20  1 21 45 23 12
20  2 23 90 12 34
30  1 44 21 45 23
      ⎕←N←'DEPT,SEX' TABULATE M
45 23 12 23
90 12 34 44

21 45 23 12
23 90 12 34

44 21 45 23
 0  0  0  0
      'DEPT M   F'⍪(CLM DEPT),+/N
DEPT M   F
 10 103 180
 20 101 159
 30 133   0
      M←39 23 12 56 33
      DEPT←'ABC' ◊ DEPTV←'AABBC'
      'DEPT,SEX' TABULATE 'DEPTV,SEXV,M'
39 23
12 56
33  0
```

Function listing:

```
     ∇ Z←X TABULATE Y;A;C;H;I;IN;J;L;N;P;R;U;V;W;G
[1]   ⍝⍝TABULATES DATA IN MATRIX Y ACCORDING TO THE K
       EYS IN X
[2]   ⍝X ←→ CHARACTER VECTOR OF K KEY VARIABLES SEPAR
       ATED BY A COMMA
[3]   ⍝Y ←→ MATRIX OF DATA THE FIRST K COLUMNS OF WHI
       CH
[4]   ⍝CORRESPOND TO THE KEY VARIABLES
[5]   ⍝THE REMAINING N COLUMNS CONTAIN THE DATA TO BE
       TABULATED
[6]   ⍝Z CONTAINS ONE DIMENSION FOR EACH KEY VARIABLE
       AND
[7]   ⍝IF N>1 AN ADDITIONAL DIMENSION OF LENGTH N
[8]   ⍝IF ONE OF THE KEY VARIABLES IS CHARACTER TYPE,
[9]   ⍝Y MUST BE A CHARACTER VECTOR WITH KEY DATA VAR
       IABLES AND
[10]  ⍝DATA VARIABLES SEPARATED FROM EACH OTHER BY A
       COMMA
[11]  ⍝NEEDS RIOTA1 SUBSUM NSS
[12]   R←U←IN←0⍴I←0
[13]   N←0∊0\0⍴Y
[14]  L1:V←',' NSS 'X',0⍴I←I+1
[15]   →E1⌈⍳2≠⎕NC V
[16]   R←R,1↑⍴A←⊥V
[17]   C←' '∊0\0⍴A
[18]   ⊥N/'→L2,0⍴G←Y[;I]'
[19]   W←',' NSS 'Y'
[20]   →E2⌈⍳2≠⎕NC W
[21]   G←⊥W
[22]   →E3⌈⍳C≠' '∊0\0⍴G
[23]  L2:→(C^2∊⍴⍴A)/L3
[24]   J←A⍳G
[25]   →L4
[26]  L3:J←A RIOTA1 G
[27]  L4:IN←IN,J
[28]   ⊥(0∊⍴U)/'U←⍳⍴J'
[29]   U←U[⍋J[U]]
[30]   →L1⌈⍳0≠⍴X
[31]   IN←((I,(⍴IN)÷I)⍴IN)[;U]
[32]   L←∨≠⍀1⌽ 0 1 ↓(IN≠⍀1⌽IN),1
[33]   P←(1↓P)-⍀1↓P←(L/⍳⍴L),1+⍴L
[34]   ⊥N/'Y←(0,I)↓Y'
[35]   ⊥(~N)/'Y←⊥Y'
[36]   Y←(2↑(⍴Y),1)⍴Y
[37]   Y←(P,1) SUBSUM Y[U;]
[38]   IN←IN[;+\P]
[39]   R←R,×/1↓⍴Y
[40]   Z←(×/R)⍴0
[41]   IN←1+R⊥⍀1+IN,[1] 1
[42]   H←1
[43]  L5:Z[IN+H-1]←Y[;H]
[44]   →L5⌈⍳(⍀1↑⍴Y)≥H←H+1
[45]   Z←((-1=⍀1↑R)↓R)⍴Z
[46]   →0
[47]  E1:'VARIABLE ',V,' NOT VALID'
[48]   →0
[49]  E2:'VARIABLE ',W,' NOT VALID'
[50]   →0
[51]  E3:'VARIABLES ',V,' AND ',W,' ARE NOT OF THE SA
       ME TYPE'
[52]  ⍝841203 13.44
     ∇
```

Name: ALFASORT - Alphabetizes matrix X

Syntax: Z←ALFASORT X

Description:

X: Character matrix
Z: Matrix X with its rows in alphabetical order. Underscored letters are treated equally with the normal ones.

Example:

```
      M←'∘'LIST'RAUHA∘LISSU LOUE∘LISSU∘RAUHA∘LISSU1'
      M
RAUHA
LISSU LOUE
LISSU
RAUHA
LISSU1
      ALFASORT M
LISSU
LISSU LOUE
LISSU1
RAUHA
RAUHA
```

Function listing:

```
    ▽ Z←ALFASORT X;⎕IO
[1]  ⍝ALPHABETIZES MATRIX X
[2]   ⎕IO←0
[3]   Z←' ABCDEFGHIJKLMNOPQRSTUVWXYZ⍙0123456789'
[4]   Z←Z,' ABCDEFGHIJKLMNOPQRSTUVWYXZ⍙'
[5]   Z←X[⍋38⊥⍉38|Z⍳X;]
[6]  ⍝840802 11.22
    ▽
```

Name: CLASSIFYX - Classifies Y in terms of vector X

Syntax: Z←X CLASSIFYX Y

Description:

X: Numeric vector of unique elements in any order
Y: Numeric array
Z: Integer array of shape ⍴Y. The elements of Y are cut down to the nearest values found in X and assigned the corresponding indices of X. If an element of Y is smaller than all elements of X, it is assigned the value ⎕IO-1.

The result is origin dependent.

Example:

```
      V←3.5 10.5 7.5
      A←2 2 8 RNDM 0 15 1
      A
 8.2  2.5  0.6  8.4  6    8.7  3.6  0.1
 8.3  1.3 11.4  1    6.3  1.8  9.5  1

14.1  2.3 10.5  0.5  9.6  1.6 14.7 12.3
10.8 14.1  2.1  8.8  0    5.1  4.7  8.8
      A1←V CLASSIFYX A
      A1
3 0 0 3 1 3 1 0
3 0 2 0 1 0 3 0

2 0 3 0 3 0 2 2
2 2 0 3 0 1 1 3
      I←I-0,1↓¯1⌽I←(V1≠1⌽V1)/⍳⍴V1←(,A1)[⍋,A1]
      I
12 5 6 9
      M←(⍕CLM V2←V[⍋V])BES' : 'BES1⍕CLM I
      'FROM 'BES1(' ',[1]⍕CLM V2)BES' TO 'BES1 M
FROM      TO  3.5 : 12
FROM  3.5 TO  7.5 :  5
FROM  7.5 TO 10.5 :  6
FROM 10.5 TO      :  9
```

Function listing:

```
     ∇ Z←X CLASSIFYX Y;B;D;I;L
[1]   ⍝⍝CLASSIFIES Y IN TERMS OF VECTOR X
[2]   ⍝Y[I] IS A MEMBER OF CLASS J IF X[J]≤Y[I]<L/(X>
       X[J])/X
[3]   ⍝Z[I] ↔ CLASS Y[I] OR ⎕IO-1 IF Y[I]<L/X
[4]    ⊥(B←~1∈⍴D←⍴Y)/'Y←,Y'
[5]    I←⍋X,Y
[6]    L←I>(⍴X)-~⎕IO
[7]    Z←(⍴Y)⍴0×2
[8]    Z[(L/I)-⍴X]←((⎕IO-1),⍋X)[⎕IO+L/+\~L]
[9]    ⊥B/'Z←D⍴Z'
[10] ⍝840802 11.15
     ∇
```

Name: FR - Finds the frequencies of Y in the classes of X

Syntax: Z←X FR Y

Description:

X: Numeric vector consisting of the lower boundary of the first class, class width, and number of classes
Y: Numeric array
Z: Integer vector of shape X[3] indicating the number of elements of Y in each class. The lower boundary of class I is X[1]+X[2]×(I-1). Y[J] is a member of the I'th class if it is between the lower boundaries of I and I+1.

Example:

```
      M←3 8 RNDM 0 10 1
      M
3.1 5.5 5.4 4.5 3   1.6 3.9 6.7
8.6 5.1 8.7 0.7 5.3 1.4 8   2.7
4.2 9.1 4.9 9.5 1.3 1.1 9   6.9
      V←0 1.5 4 FR M
      V
4 2 4 6
      M←' - 'BES1(⍕CLM V←1.5×⍳4)BES' : 'BES1⍕CLM V
      'CLASS'BES1(⍕CLM⍳4),' → 'BES1(⍕CLM V-1.5)BES M
CLASS1 → 0   - 1.5 : 4
CLASS2 → 1.5 - 3   : 2
CLASS3 → 3   - 4.5 : 4
CLASS4 → 4.5 - 6   : 6
```

Function listing:

```
    ∇ Z←X FR Y;⎕IO
[1]  ⍝⍝FINDS THE FREQUENCIES OF Y IN THE CLASSES OF
      X
[2]  ⍝X ↔ THE LOWER BOUNDARY OF THE FIRST CLASS, CL
     ASS WIDTH, NUMBER OF CLASSES
[3]  ⍝Z[I] GIVES THE NUMBER OF ELEMENTS OF Y IN EACH
       CLASS
[4]   ⎕IO←0
[5]   Z←+/(⍳X[2])∘.=⌊((,Y)-X[0])÷X[1]
[6]  ⍝840802 11.20
    ∇
```

Name: ORDER - Index vector that orders matrix Y in X collating order

Syntax: Z←X ORDER Y

Description:

X: Vector or matrix
Y: Matrix of the same type as X
Z: Index vector that orders Y by rows in X collating order. If X is a matrix its columns idicate equal characters.

The result is origin dependent.

Example:

```
      ⎕←M←?5 4⍴3
3 3 3 2
2 1 1 1
1 3 2 3
1 1 1 3
3 2 3 2
      ⎕←V←(⍳3)ORDER M
4 3 2 5 1
      M[V;]
1 1 1 3
1 3 2 3
2 1 1 1
3 2 3 2
3 3 3 2
      ⎕←N←3 4⍴'ABCD1234ABCD'
ABCD
1234
ABCD
      ⎕←M←' 'LIST'BERIT CYNTHIA 1CYNTHIA DOUG D4'
BERIT
CYNTHIA
1CYNTHIA
DOUG
D4
      ⎕IO←0
      ⎕←V←N ORDER M
2 0 1 4 3
      M[V;]
1CYNTHIA
BERIT
CYNTHIA
D4
DOUG
```

Function listing:

```
     ∇ Z←X ORDER Y;C;M;O;P
[1]   ⍝⍝INDEX VECTOR THAT ORDERS MATRIX Y IN X COLLAT
       ING ORDER
[2]   ⍝X IS A VECTOR OR A MATRIX, WHOSE COLUMNS INDIC
       ATE EQUAL CHARACTERS.
[3]    O←⎕IO
[4]    Z←⍳1⌷⍴Y
[5]    P←¯1↑⍴X
[6]    X←,X
[7]    C←⍳1↓⍴Y
[8]   L:M←X⍳Y[Z;(-(⍴C)⌊10)↑C]
[9]    Z←Z[⍋(P+1)⊥⍉(P⍴(-O)+M)+P×M=O+⍴X]
[10]   →(0≠⍴C←¯10↓C)/L
[11] ⍝840802 11.23
     ∇
```

Name: AY - Sets true the first ones in groups of ones

Syntax: Z←AY X

Description:

X: Logical array
Z: Logical array of shape ⍴X. Evaluated along the last dimension, only the first ones in groups of ones in X are set true.

Example:

```
      ⎕←M←4 10⍴1 1 1 1 0 0 0 1 1 1
1 1 1 1 0 0 0 1 1 1
1 1 1 0 0 0 1 1 1 1
1 1 0 0 0 1 1 1 1 1
1 0 0 0 1 1 1 1 1 1
      AY M
1 0 0 0 0 0 0 1 0 0
1 0 0 0 0 0 1 0 0 0
1 0 0 0 0 1 0 0 0 0
1 0 0 0 1 0 0 0 0 0
```

Function listing:

```
    ∇ Z←AY X
[1]   ⍝SETS TRUE THE FIRST ONES IN GROUPS OF ONES
[2]     Z←X>((-⍴⍴X)↑¯1)↓0,X
[3]   ⍝840802 13.25
    ∇
```

Name: GRPV - Groups of ones in vector Y marked by vector X

Syntax: Z←X GRPV Y

Description:

X: Logical scalar or a vector of the same length as Y
Y: Logical scalar or vector
Z: Logical vector of the same length as Y. Those groups of ones in Y that are marked by ones in corresponding positions in X are preserved. Other groups of ones are set untrue.

A group in Y is marked by X by having one or a multiple of ones somewhere in the range of the corresponding group of Y. If X is a scalar, it is handled as if it were a vector of all ones/zeroes.

Example:

```
      0 1 0 1 0 1 1 0 0 0 GRPV V← 1 1 1 0 0 1 1 0 1 1
      1 GRPV V
1 1 1 0 0 1 1 0 1 1
      V←'1,23 320 2,90 0,01 12'
      3 4⍴(⍞←(V=',')GRPV V≠' ')/V
1 1 1 1 0 0 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0
1,23
2,90
0,01
```

Function listing:

```
    ▽ Z←X GRPV Y;⎕IO;A
[1]   ⍝GROUPS OF ONES IN VECTOR Y MARKED BY VECTOR X
[2]    ⎕IO←0
[3]    Z←(1+¯1↑A←+\Y)¯1↓0,Y)⍴0
[4]    Z[(X^Y)/A]←1
[5]    Z←Y^Z[A]
[6]   ⍝840802 13.41
    ▽
```

Name: LEN - Lengths of groups of ones in vector X

Syntax: Z←LEN X

Description:

X: Logical vector
Z: Integer vector returning the lengths of groups of ones in X.

Example:

```
      LEN 1 1 1 0 1 0 0 0 1 1 1 1 1
3 1 5
```

Function listing:

```
    ∇ Z←LEN X;I
[1]  ⍝LENGTHS OF GROUPS OF ONES IN VECTOR X
[2]   Z←(Z>0)/Z←(1↓I)-1+¯1↓I←(~X)/⍳⍴X←0,X,0
[3]  ⍝840802 13.45
    ∇
```

Name: LOOG - Logical vector of length X with indices indicated by pairs in Y set true

Syntax: L←X LOOG Y

Description:

X: Positive integer
Y: Non-negative integer array consisting of pairs whose elements are ≤X.
Z: Logical vector of length X. Elements between the indices given in the pairs of Y, inclusive, are set true, and the other elements are set untrue.

The result is origin dependent.

Example:

```
      12 LOOG 2 2 5 8 10 11
0 1 0 0 1 1 1 1 0 1 1 0
      ⎕IO←0
      12 LOOG 1 1 4 7 9 10
0 1 0 0 1 1 1 1 0 1 1 0
```

Function listing:

```
    ∇ L←X LOOG Y;K
[1]   ⍝LOGICAL VECTOR OF LENGTH X WITH INDICES INDICA
      TED BY PAIRS IN Y SET TRUE
[2]   L←X⍴0
[3]   Y←((0.5××/⍴Y),2)⍴Y
[4]   K←0≠-/Y
[5]   L[K/Y]←1
[6]   L←L∨≠\L
[7]   L[(~K)/Y]←1
[8]   ⍝840802 13.46
    ∇
```

Name: LY - Sets true the last ones in groups of ones

Syntax: Z←LY X

Description:

X: Logical array
Z: Logical array of shape ⍴X. Evaluated along the last dimension, only the last ones in groups of ones in X are set true.

Example:

```
      ⎕←M←4 10⍴1 1 1 1 0 0 0 1 1 1
1 1 1 1 0 0 0 1 1 1
1 1 1 0 0 0 1 1 1 1
1 1 0 0 0 1 1 1 1 1
1 0 0 0 1 1 1 1 1 1
      LY M
0 0 0 1 0 0 0 0 0 1
0 0 1 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 1
```

Function listing:

```
    ∇ Z←LY X
[1]  ⍝SETS TRUE THE LAST ONES IN GROUPS OF ONES
[2]   Z←X>((-⍴⍴X)↑1)↓X,0
[3]  ⍝840802 13.48
    ∇
```

Name: COMB - Combinations of integers from 1 to Y in length X

Syntax: Z←X COMB Y

Description:

X: Positive integer
Y: Positive integer. Y ≥ X.
Z: Integer matrix. On each row of Z there is a different combination of integers from range 1 to Y. The number of elements in the combinations is given in X.

This function needs the auxiliary functions RHO and PIOTA.

Example:

```
      3 COMB 5
1 2 3
1 2 4
1 2 5
1 3 4
1 3 5
1 4 5
2 3 4
2 3 5
2 4 5
3 4 5
```

Function listing:

```
     ∇ Z←X COMB Y;H;I;D;B;⎕IO
[1]   ⍝⍝COMBINATIONS OF INTEGERS FROM 1 TO Y IN LENGT
      H X
[2]   ⍝NEEDS RHO PIOTA
[3]    I←⍳D←1+Y-X+H←⎕IO←0
[4]    B←⌊I∘(∘I)∘.!∘⍳Y
[5]    Z←((X!Y),X)⍴I←0
[6]   R1:Z[;H]←B[I;H] RHO(1+H+⍳D)[I←I PIOTA D-I]
[7]    →(X>H←H+1)/R1
[8]   ⍝840802 13.28
     ∇
```

Name: COMBI - Index matrix with combinations of indices to X and Y when the Keys match

Syntax: Z←X COMBI Y

Description:

X: Integer vector
Y: Integer vector
Z: 2-column index matrix containing all the combinations of indices to Key vectors X and Y when the Keys match.

Applied to the arguments, each combination returns two identical integers. This function uses the auxiliary functions FIELDS,RHO,PIOTA,and EPS.

Example:

```
      ⎕←M←(V1←21 6 9 21 9)COMBI V2←9 21 32 21 9 9
3 1
3 5
3 6
5 1
5 5
5 6
1 2
1 4
4 2
4 4
      V1[M[;1]]UPON V2[M[;2]]
 9  9  9  9  9  9 21 21 21 21
 9  9  9  9  9  9 21 21 21 21
```

Function listing:

```
    ∇ Z←X COMBI Y;LA;LB;FA;FB;⎕IO;F;UA;UB
[1]   ⍝⍝INDEX MATRIX WITH COMBINATIONS OF INDICES TO
        X AND Y WHEN THE KEYS MATCH
[2]   ⍝X AND Y CONTAIN NUMERIC KEYS
[3]   ⍝NEEDS FIELDS RHO PIOTA EPS
[4]    ⎕IO←1
[5]    X←(LA←X EPS Y)/X
[6]    Y←(LB←Y EPS X)/Y
[7]    UA←⍋X
[8]    UB←⍋Y
[9]    FA←FIELDS X[UA]
[10]   FB←FIELDS Y[UB]
[11]   F←FA×FB
[12]   Z←1+⍳(¯1 PIOTA F)÷F RHO⌈F÷FA
[13]   Z←Z,[1.1] 1+(F RHO FB)|¯1 PIOTA F
[14]   Z←Z+(F RHO+\¯1↓0,FA),[1.1] F RHO+\¯1↓0,FB
[15]   Z←(LA/⍳⍴LA)[UA[Z[;1]]],(LB/⍳⍴LB)[UB[Z[;,2]]]
[16] ⍝840802 13.32
    ∇
```

Name: EGET - Exclusive indexing X[Y]

Syntax: Z←X EGET Y

Description:

X: Any array
Y: Integer matrix (vector for a vector argument X).
1↑⍴Y must be equal to ⍴⍴X.
Z: Vector returning the elements of X that are indexed by Y. Each column of Y is interpreted as a set of indices applying to a particular element of X.

Example:

```
      ⎕←A←2 3 7⍴'ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
ABCDEFG
HIJKLMN
OPQRSTU

VWXYZ12
3456789
0ABCDEF
      ⎕←M←3 5⍴1 1 1 2 2 1 3 2 2 2 6 7 7 6 2
1 1 1 2 2
1 3 2 2 2
6 7 7 6 2
      A EGET M
FUN84
```

Function listing:

```
    ∇ Z←X EGET Y
[1]   ⍝⍝EXCLUSIVE INDEXING X[Y]
[2]   ⍝1↑⍴Y ←→ ⍴⍴X
[3]    Z←(,X)[⎕IO+(⍴X)⊥Y-⎕IO]
[4]   ⍝840802 13.17
    ∇
```

Name: EPUT - Exclusive put

Syntax: Z←X EPUT Y

Description:

X: Any array
Y: Array of type X. There must be as many elements in Y as there are columns in global matrix M - unless Y is a scalar.
Z: Argument X with elements indicated by exclusive indices in global variable M replaced by the element(s) of Y.

Each column of the global matrix M represents a set of indices applying to a particular element of X. This function can be used with the auxiliary function IN.

Example:

```
      ⎕←M←4 8⍴⍳32
 1  2  3  4  5  6  7  8
 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24
25 26 27 28 29 30 31 32
      V←10 200 3000
      ⎕←M←2 3⍴⍳6
1 2 3
4 5 6
      ⎕←M←M EPUT V
   1    2    3   10    5    6    7    8
   9   10   11   12  200   14   15   16
  17   18   19   20   21 3000   23   24
  25   26   27   28   29   30   31   32
      M EPUT 0 IN ⎕←2 4⍴⍳4
1 2 3 4
1 2 3 4
   0    2    3   10    5    6    7    8
   9    0   11   12  200   14   15   16
  17   18    0   20   21 3000   23   24
  25   26   27    0   29   30   31   32
```

Function listing:

```
    ∇ Z←X EPUT Y
[1]  ⍝⍝EXCLUSIVE PUT
[2]  ⍝PUTS Y TO LOCATIONS IN X GIVEN BY EXCLUSIVE IN
      DICES IN GLOBAL VARIABLE M
[3]   Z←,X
[4]   X←⍴X
[5]   Z[⎕IO+X⊥M-⎕IO]←,Y
[6]   Z←X⍴Z
[7]  ⍝840802 13.20
    ∇
```

Name: FHEADER - Formats text Y to fields of length X[1;]

Syntax: Z←X FHEADER Y

Description:

X: Non-negative integer scalar, vector, or one- or two-row matrix. The second row must have 0's, 1's and 2's only.
Y: Character vector or matrix. 1≠Y ↔ separator character other than a blank. Each row may contain separator characters.
Z: If Y is a matrix, each row of Y is formatted to a field according to instructions given in the corresponding column of X. If Y is a vector, each string between adjacent separators is formatted to a field. X[1;] gives the lengths of the fields and X[2;] the way the text is justified in the field: 0=left, 1=right, 2=center. In the case of matrix Y each string between the separator characters forms an individual row in the field.

Character ∘ or a blank may be used as a fill character for not having the rows left justified. In absence of a second row of X, all the texts will be left justified in the fields.

Example:

```
      5 10 10 FHEADER '∆ABCDEFG∆ABC∆ABC'
ABCDEABC        ABC
      □←F←2 3⍴10 20 10 0 2 1
10 20 10
 0  2  1
      H←'∩LEFT ∩TEXT∩ PRINTS∩HERE'
      H←H UPON 'CENTER∩TEXT∩∘∘HERE'
      □←H←H UPON 'RIGHT∩HEADER∩HERE'
∩LEFT ∩TEXT∩ PRINTS∩HERE
CENTER∩TEXT∩∘∘HERE
RIGHT∩HEADER∩HERE
      (⍳AXIS 40)UPON F FHEADER H
0000000001111111111222222222233333333334
1234567890123456789012345678901234567890
LEFT              CENTER              RIGHT
TEXT              TEXT                HEADER
 PRINTS             HERE              HERE
HERE
```

Function listing:

```
     ∇ Z←X FHEADER Y;A;C;J;L;M;N;P;R;S;W
[1]   ⍝⍝FORMATS TEXT Y TO FIELDS OF LENGTH X[1;]
[2]   ⍝X[2;] ↔ 0=JUSTIFY LEFT, 1=JUSTIFY RIGHT, 2=JU
      STIFY CENTERED
[3]   ⍝(1⍴Y) ↔ SEPARATOR CHARACTER OTHER THAN A BLAN
      K
[4]   ⍝Y ↔ VECTOR OR MATRIX OF HEADERS
[5]   ⍝EACH ROW MAY CONTAIN SEPARATOR CHARACTERS
[6]   ⍝CHARACTER ∘ MAY BE USED AS A FILL CHARACTER (N
      OT JUSTIFIED)
[7]   ⍝NEEDS LISTO CHNG1
[8]    X←(L←X[1;]≠0)/X←((¯2↑ 1 1 ,⍴X)⍴X),[1] 0
[9]    Z←(0,+/W←X[1;])⍴' '
[10]   →0⊥∨0∈⍴Y
[11]   S←''⍴1⍴Y
[12]   ⊥(2∈⍴⍴Y)/'Y←0 1↓(Y[;1]=S)⌽'' '',Y'
[13]   ⊥(A←1∈⍴⍴Y)/'Y←S LISTO '' ∘''CHNG1 1↓Y'
[14]   Y←L⌿Y
[15]   →A/R0,0⍴Y←((⍴Y),1)[1 3 2]⍴Y
[16]   →R0⌈∨1^.=N←1+,Y+.=S
[17]   Y←(,N∘.≥∨M←⌈/N)⌿S LISTO Y
[18]   Y←(((1↑⍴Y)÷M),M,¯1↑⍴Y)⍴Y
[19] R0:L←Y≠' '
[20]   →R1⌈∨0∈⍴A←A/∨⍴A←×10⊥J←X[2;]
[21]   L[A;;]← 2 1 3 ⍉((⍴A),1↓⍴L)[2 1 3]⍴∨/[2] L[A;;]
[22] R1:P←L⌈.×∨¯1↑⍴Y
[23]   Y←((¯1↓⍴Y),C←⌈/W)↑Y
[24]   R←(⍉A⍴J≠0)×-⌈(⍉A⍴0.5*2=J←10⊥J)×0⌈(⍉(A←⌽⍴P)⍴W)-
       P
[25]   Y←R⌽Y
[26]   Z←'∘ ' CHNG1(,W∘.≥∨C)/(×≠ 2 2 ⍴1,⍴Y)⍴ 2 1 3 ⍉Y
[27] ⍝840802 13.11
     ∇
```

Name: FMT - X ⍕ Y where zero elements of Y are not printed

Syntax: Z←X FMT Y

Description:

X: Integer vector
Y: Numeric array
Z: Z is formed equally with the dyadic primitive function ⍕ with the difference of not printing the zero elements of Y.

This function needs the auxiliary function RHO.

Example:

```
      ⎕←M←3 5 RNDM ¯3 2
 0 ¯2 ¯2 ¯3  2
 0  0 ¯3 ¯3  2
 0 ¯3 ¯1 ¯3  1
      10 2 FMT M
          ¯2.00     ¯2.00     ¯3.00      2.00
                    ¯3.00     ¯3.00      2.00
          ¯3.00     ¯1.00     ¯3.00      1.00
```

Function listing:

```
    ∇ Z←X FMT Y;I;L;W;⎕IO
[1]  ⍝⍝X⍕Y WHERE ZERO ELEMENTS OF Y ARE NOT PRINTED
[2]  ⍝NEEDS RHO
[3]   ⎕IO←1
[4]   X←(2×¯1↑⍴Y)⍴X
[5]   W←0≠((×/¯1↓⍴Y),¯1↑1,⍴Y)⍴Y
[6]   →R1⌈⍳∧/I=1↑I←X[¯1+2×⍳0.5×⍴X]
[7]   L←,W[;I RHO⍳⍴I]
[8]   Z←((¯1↓⍴Y),+/I)⍴L\L/,X⍕Y
[9]   →0
[10] R1:Z←,X⍕Y
[11]  Z[(X×(~,W)/⍳×/⍴W)∘.+(⍳X)-X←1↑I]←' '
[12]  Z←((¯1↓⍴Y),+/I)⍴Z
[13] ⍝840802 13.02
    ∇
```

Name: FMT⍠ - X ⍕ Y with '.' inserted bw thousands and '.' → ','

Syntax: Z←X FMT⍠ Y

Description:

X: Integer vector
Y: Numeric array
Z: Character array where elements of Y are formed equally with the dyadic primitive function ⍕. A point is inserted to separate thousands and the decimal point is replaced by a comma.

Example:

```
      8 2 5 0 FMT⍠ ¯2.999 ¯3.666
¯3,00   ¯4
      ⎕←M←3 2 RND⍠ (1E6×2 6),4
4981621.63  2362444.412
5039494.378 5272351.864
5182043.556 3737372.267
      15 2 FMT⍠ M
   4.981.621,63   2.362.444,41
   5.039.494,38   5.272.351,86
   5.182.043,56   3.737.372,27
```

Function listing:

```
     ∇ Z←X FMTE Y;DX;K;SK;L;D;I;J;⎕IO
[1]   ⍝X F Y WITH '.' INSERTED BW THOUSANDS AND ',' →
        ','
[2]    ⎕IO←1
[3]    X←(2*¯1↑1,⍴Y)⍴X
[4]    DX←⍴Y←X⍕Y
[5]    Y←(¯2↑1,⍴Y)⍴Y
[6]    X←((0.5*⍴X),2)⍴X
[7]    SK←+\K←X[;1]
[8]    L←0≠D←X[;2]
[9]    Y[;L/SK-D]←','
[10]   I←K-D+L
[11]   J←(¯1+4-4|I)∘.+Z←⍳⌈/K,0
[12]   J←(0≠4|J)∨(I∘.<Z)
[13]   J←(×J)× 0 1 ↓+\((+/~J)+¯1↓0,SK),J
[14]   J←(,K∘.≥Z)/,J
[15]   Y←,('.',Y)[;J+1]
[16]   L←(⍴Y)⍴(⍳+/K)∈1+0,SK
[17]   J←¯1+((~L)^Y=',')/⍳⍴L
[18]   Y[(Y[J]=' ')/J]←'0'
[19]   J←((~L[J])^Y[J]='¯')/J
[20]   Y[J∘.- 1 0]←((⍴J),2)⍴'¯0'
[21]   Y[((Y=',')^L∨¯1⌽Y=' ')/⍳⍴L]←' '
[22]   J←((Y='.')^L∨¯1⌽Y='¯')/⍳⍴L
[23]   Y[J∘.- 1 0]←((⍴J),2)⍴' ¯'
[24]   Z←DX⍴Y
[25]  ⍝840802 13.05
     ∇
```

Name: LIMITE - Limits numbers in Y to format X (width, decimals)

Syntax: Z←F LIMITE A

Description:

X: Integer vector whose length is twice the length of the last axis of Y - or a single pair
Y: Numeric array
Z: The elements of Y reshaped to fit the format X. If an element does not fit to this format it is replaced by the nearest number that meets the requirements of X. E.g. 5 2 LIMITE V limits V to values between ¯9.99 and 99.99.

The arguments have such a relation as in the case of the dyadic function format (⍕). This function is used in connection with the dyadic function format.

Example:

```
      M←2 3⍴23 2.2 12.34 3.456 0.21 0.054
      M
23     2.2   12.34
 3.456 0.21   0.054
      3 1⍕M
***2.2***
3.50.20.1
      M←3 1 LIMITE M
      3 1⍕M
9.92.29.9
3.50.20.1
      V⍕(V←5 2 6 2 5 1 6 1)LIMITE 100 100 ¯200 ¯200
```

Function listing:

```
    ∇ Z←F LIMITE A;⎕IO;D
[1]   ⍝LIMITS NUMBERS IN A TO FORMAT F (WIDTH, DECIMA
      LS)
[2]   D←0⍴⎕IO←1
[3]   ⍎(2≠⍴F)/'F←((D←⍴A),2)⍴F'
[4]   Z←-/[1] 10*(¯1 ¯2 ∘.+-/F+×F),[0.1]-(2,D)⍴ 0 1
      /F
[5]   ⍎(0∧.=, 0 1 /F)/'Z←⌈Z'
[6]   Z←(D⍴ 1 0 ≠Z)⌊(-D⍴ 0 1 ≠Z)⌈A
[7]   ⍝840801 08.45
    ∇
```

Name: FIELDS - Lengths of fields of same character or number in vector X

Syntax: Z←FIELDS X

Description:

X: Any vector
Z: Integer vector returning the lengths of fields of the same character or number in X.

Example:

```
      FIELDS'AAABB CCCCD'
3 2 1 4 1
```

Function listing:

```
     ▽ Z←FIELDS X;L;I
[1]   ⍝LENGTHS OF FIELDS OF SAME CHARACTER OR NUMBER
      IN VECTOR X
[2]   →(0∊⍴X)/⍴Z←⍳0
[3]   Z←(1↓I)-¯1↓I←(L/⍳⍴L),⎕IO+⍴L←1,(1↓X)≠¯1↓X
[4]   ⍝840802 13.37
     ▽
```

Name: FOG - First of groups

Syntax: Z←FOG X

Description:

X: Any array
Z: Logical array of shape ⍴X. 1 indicates the beginning of a group in X, evaluated along the last dimension.

Example:

```
      ⎕←M←3 6⍴3 3 3 4 4
3 3 3 4 4 3
3 3 4 4 3 3
3 4 4 3 3 3
      ⎕←N←FOG M
1 0 0 1 0 1
1 0 1 0 1 0
1 1 0 1 0 0
      M×N
3 0 0 4 0 3
3 0 4 0 3 0
3 4 0 3 0 0
```

Function listing:

```
    ∇ Z←FOG X
[1]  ⍝⍝FIRST OF GROUPS
[2]  ⍝Z IS A LOGICAL ARRAY WHERE 1 INDICATES THE BEG
      INNING OF A GROUP IN X
[3]   →(0∈⍴X)/⍴Z←⍳0
[4]   Z←1,((Z↑¯1)↓X)≠((Z←-1⌈⍴⍴X)↑1)↓X
[5]  ⍝840802 13.39
    ∇
```

Name: PARTSUM - Sums over fields in Y when X contains the field widths

Syntax: Z←X PARTSUM Y

Description:

X: Positive integer vector or scalar. +/X must be ≤⍴Y.
Y: Numeric vector
Z: Sums over the fields in Y. The field widths are given in X.

Example:

```
      4 3 2 1 PARTSUM 1 2 3 4 5 6 7 8 9 10
10 18 17 10
      ⍴⎕←5 PARTSUM ⍳10
15
1
```

Function listing:

```
    ∇ Z←X PARTSUM Y;⎕IO
[1]   ⍝SUMS OVER FIELDS IN Y WHEN X CONTAINS THE FIEL
      D WIDTHS
[2]   ⎕IO←1
[3]   Z←Z-¯1↓0,Z←(+\Y)[+\X]
[4]   ⍝840802 13.50
    ∇
```

Name: PIND - Vector X RHO ⍴X

Syntax: Z←PIND X

Description:

X: Vector of non-negative elements
Z: Vector X RHO ⍴X, where RHO is another auxiliary function. In other words, Z is a vector (X[⎕IO]⍴⎕IO),(X[⎕IO+1]⍴⎕IO+1),...

The result is origin dependent.

Example:

```
      ⎕IO←0
      PIND 3 0 4 4
0 0 0 2 2 2 2 3 3 3 3
```

Function listing:

```
    ∇ Z←PIND X;L
[1]   ⍝VECTOR X RHO ⍴X
[2]   Z←(+/X)⍴0
[3]   Z[+\¯1↓⎕IO,L/X]←(L←⍳X)/⍴X
[4]   Z←⌈\Z
[5]   ⍝840802 13.57
    ∇
```

Name: PINI - Vector of fields formed by index generating elements of X

Syntax: Z←PINI X

Description:

X: Non-negative integer scalar or vector
Z: Integer vector with fields formed by index generating each element of X.

PINI could be regarded as monadic vector ⍳.
The result is origin dependent.

Example:

```
      PINI 6 4 0 3
1 2 3 4 5 6 1 2 3 4 1 2 3
```

Function listing:

```
    ∇ Z←PINI X
[1]  ⍝⍝VECTOR OF FIELDS FORMED BY INDEX GENERATING E
      LEMENTS OF X
[2]  ⍝MONADIC VECTOR ⍳
[3]   Z←(+/X←(X≠0)/X)⍴1
[4]   Z[+\¯1↓⎕IO,X]←1-¯1↓(~⎕IO),X
[5]   Z←+\Z
[6]  ⍝840802 14.00
    ∇
```

Name: PIOTΔ - Vector (X[1]+\Y[1]), (X[2]+\Y[2]),...

Syntax: Z←X PIOTΔ Y

Description:

X: Numeric vector or scalar
Y: Non-negative integer vector or scalar.
If X is a vector ⍴Y must be equal to ⍴X.
Z: Numeric vector with fields that are formed by index generating each element of Y and then adding the corresponding element of X to each field.

PIOTΔ could be regarded as monadic vector \ with index origins in the left argument. If X or Y is a scalar it is extended to conform with the other argument. The result is origin dependent.

Example:

```
      ⎕IO←0
      10 PIOTΔ 4 5 6
10 11 12 13 10 11 12 13 14 10 11 12 13 14 15
      ¯9.5 ¯6.5 ¯3.5 PIOTΔ 6 3 1
¯9.5 ¯8.5 ¯7.5 ¯6.5 ¯5.5 ¯4.5 ¯6.5 ¯5.5 ¯4.5 ¯3.5
```

Function listing:

```
    ▽ Z←X PIOTΔ Y
[1]  ⍝VECTOR (X[1]+\Y[1]), (X[2]+\Y[2]),...
[2]  ⍝X OR Y CAN BE A SCALAR,TOO
[3]   X←(Z←Y≠0)/X
[4]   Z←(+/Y←(⍴X)⍴Z/Y)⍴2-1
[5]   Z[+\¯1↓⎕IO,Y]←(X+1)-¯1↓(~⎕IO),Y+X
[6]   Z←+\Z
[7]  ⍝840802 14.04
    ▽
```

Name: RHO - Vector (X[1]⍴Y[1]), (X[2]⍴Y[2]),...

Syntax: Z←X RHO Y

Description:

X: Non-negative integer vector or scalar
Y: Any array. If X is a vector ⍴Y must equal ⍴X.
Z: Numeric vector formed by reshaping every element of (,Y) by the corresponding element of X, and by catenating the results into one vector.

If X or Y is a scalar, it is extended to conform with the other element.

Example:

```
      3 RHO 0 ¯2 .5
0 0 0 ¯2 ¯2 ¯2 0.5 0.5 0.5
      3 6 4 RHO 'ABC'
AAABBBBBBCCCC
```

Function listing:

```
    ∇ Z←X RHO Y;⎕IO
[1]  ⍝VECTOR (X[1]⍴Y[1]), (X[2]⍴Y[2]),...
[2]   ⎕IO←0
[3]   Z←(+/X←(⍴Y←Z/,Y)⍴(Z←X≠0)/X)⍴0
[4]   Z[+\¯1↓X]←1
[5]   Z←Y[+\Z]
[6]  ⍝840802 14.08
    ∇
```

Name: SUBSUM - Subtotals of array Y

Syntax: Z←X SUBSUM Y

Description:

X: Non-negative integer vector. +/¯1↓X must be ≤ (⍴Y)[¯1↑X].
Y: Numeric array (not a scalar)
Z: Numeric array containing subtotals of array Y summed along the dimension given in ¯1↑X. ¯1↓X gives the lengths of the segments to be summed in Y.

Example:

```
      M←8 4 RNDM 9 900 1
      M1←(3 EXPND 3 3 2)\M
      M1[5 11 16;]←⍬←3 3 2 1 SUBSUM M
1820.5 1588.3 1369.8 1711
1776.4 1092.7 1399.4 1552.3
 860.1 1110.3 1177.2  632.1
1820.5 1588.3 1369.8 1711
1776.4 1092.7 1399.4 1552.3
 860.1 1110.3 1177.2  632.1
      M1←(1 EXPND 2 2)\M1
      ⍬←M←2 2 2 SUBSUM M
1685   1186
1068.7  866.4
 655.1 1028.4
 646.3 1240.3
1196    313.3
1026.8 1398.1
 503.9  851.5
1466.5  957.8
      M1[;3 6]←(3 EXPND 3 3 2)\M
      M1←8 2 FMT M1 ◊ M1[4 10 15;]←'='
      M1[6 12 17;]←'-' ◊ M1[;17 25 41]←'|'
      M1,'|'
  835.20  849.80|1685.00| 879.00  307.00|1186.00|
  805.20  263.50|1068.70| 180.00  686.40| 866.40|
  180.10  475.00| 655.10| 310.80  717.60|1028.40|
================|=======|================|=======|
 1820.50 1588.30|       |1369.80 1711.00|       |
----------------|-------|----------------|-------|
  495.50  150.80| 646.30| 394.30  846.00|1240.30|
  786.20  409.80|1196.00| 279.10   34.20| 313.30|
  494.70  532.10|1026.80| 726.00  672.10|1398.10|
================|=======|================|=======|
 1776.40 1092.70|       |1399.40 1552.30|       |
----------------|-------|----------------|-------|
  246.70  257.20| 503.90| 455.60  395.90| 851.50|
  613.40  853.10|1466.50| 721.60  236.20| 957.80|
================|=======|================|=======|
  860.10 1110.30|       |1177.20  632.10|       |
----------------|-------|----------------|-------|
```

Function listing:

```
     ∇ Z←X SUBSUM Y;A
[1]   ⍝SUBTOTALS OF ARRAY Y
[2]   ⍝¯1↓X ↔ LENGTHS OF THE SEGMENTS IN Y TO BE SUM
       MED
[3]   ⍝¯1↑X ↔ DIMENSION OF SUMMING
[4]    A←⎕IO-Z←¯1↑X
[5]    X←(⎕IO-1)++\¯1↓X
[6]    Y←⊥'(+\[Z]Y)[',(A⍴'X',(¯1+⍴⍴Y)⍴';'),']'
[7]    Z←Y-(A⍴(⍴⍴Y)↑¯1)↓0,[Z] Y
[8]   ⍝840802 14.17
     ∇
```

Name: DAYDIF - Number of actual days between dates Y and X

Syntax: Z←X DAYDIF Y

Description:

X: Positive integer array with its elements in form yymmdd where yy represents years, mm months, and dd days.
Y: Similar array of the same shape - or a scalar. The elements of Y must be smaller than X.
Z: Integer array of shape ⍴X or ⍴Y. Z shows the number of actual days between dates in Y and X in the form of the arguments.

This function applies to a scalar and any array accordingly with the primitive function -.

Example:

```
      850101 DAYDIF 840101 830101
366 731
      850301 840301 DAYDIF 850201 840201
28 29
```

Function listing:

```
    ∇ Z←X DAYDIF Y;D;V;YA;YB;N;⎕IO
[1]  ⍝⍝NUMBER OF ACTUAL DAYS BETWEEN DATES Y AND X
[2]  ⍝Y MUST BE < X
[3]   YA←(Y←⌊(3⍴100)⊤,Y)[⎕IO←1;]
[4]   YB←(X←⌊(3⍴100)⊤,X)[1;]
[5]   V←YB-YA
[6]   D←⌈0.25×0⌈V-4|-YA
[7]   N←0,+\ 31 28 ,10⍴5⍴ 31 30
[8]   Y←Y[3;]+N[YA]+(2<YA←Y[2;])^0=4|YA
[9]   X←X[3;]+N[YB]+(2<YB←X[2;])^0=4|YB
[10]  Z←X+(D+V×365)-Y
[11] ⍝840802 11.27
    ∇
```

Name: DAYS - Number of days in months X

Syntax: Z←DAYS X

Description:

X: Array of positive integer elements of form yymm
where yy represents years and mm months
Z: Integer array of the shape of X indicating
how many days there are in corresponding months
of X

Example:

```
      ⎕←M←2 3⍴8400+⍳6
8401 8402 8403
8404 8405 8406
      DAYS M
31 29 31
30 31 30
```

Function listing:

```
    ∇ Z←DAYS X;⎕IO
[1]   ⍝⍝NUMBER OF DAYS IN MONTHS X
[2]   ⍝X CONTAINS MONTHS IN FORM YYMM
[3]    ⎕IO←1
[4]    Z←⍴X
[5]    X← 100 100 ⊤,X
[6]    Z←Z⍴(12⍴7⍴ 31 30)[X[2;]]-(X[2;]=2)×(0≠4|X[1;])
       +1
[7]   ⍝840802 11.35
    ∇
```

Name: FMTYMD - Formats dates of form yymmdd in vector X as dd.mm.19yy

Syntax: Z←FMTYMD X

Description:

X: Integer array with its elements in form yymmdd where yy represents years, mm months, and dd days

Z: Character matrix of ⍴,X rows. Each element of X is formatted in form dd.mm.19yy in an individual row of Z. In case of X being a scalar, the result Z is a character vector.

This function can be used for years from 1910 to 1999.

Example:

```
      V←410621 840101 991231
      FMTYMD V
21.06.1941
01.01.1984
31.12.1999
      FMTYMD TODAY
03.12.1984
```

Function listing:

```
    ∇ Z←FMTYMD X
[1]  ⍝FORMATS DATES OF FORM YYMMDD IN VECTOR X AS DD
      .MM.19YY
[2]   Z←((⍴,X),1)⍴X
[3]   Z←('.','1','9', 6 0 ⍉Z)[; 7 8 0 5 6 0 1 2 3 4
      +⎕IO]
[4]   ⊥(0=⍴⍴X)/'Z←,Z'
[5]  ⍝840802 11.37
    ∇
```

Name: SIDATE - Date in the SI standard form

Syntax: Z←SIDATE

Description:

Z: Date in the SI standard form 19yy-mm-dd where yy represents years, mm months, and dd days.
A character vector.

Example:

```
      SIDATE
1984-12-03
```

Function listing:

```
     ▽ Z←SIDATE
[1]   ⍝DATE IN THE SI STANDARD FORM
[2]    Z←3↑⎕TS
[3]    ⊥(3 0 ^.=1↓Z)/'Z[1+⍳2]←2 29'
[4]    Z←'19',1↓,'-', 1 0 ⍕⍤ 10 10 ⍉Z
[5]   ⍝840802 11.42
     ▽
```

Name: TODAY - Numeric date in form yymmdd

Syntax: Z←TODAY

Description:

Z: Numeric date in form yymmdd where yy represents years, mm months, and dd days. Z is an integer.

Example:

```
      TODAY
841203
      FMTYMD TODAY
03.12.1984
```

Function listing:

```
    ▽ Z←TODAY
[1]   ⍝NUMERIC DATE IN FORM YYMMDD
[2]    Z←3↑⎕TS
[3]    ⊥(3 0 ^.=1↓Z)/'Z[1+⍳2]←2 29'
[4]    Z←100⊥100|Z
[5]   ⍝8002291022▽
[6]   ⍝840802 11.42
    ▽
```

Name: WEEKDAY - Index of the days of the week of the dates yymmdd

Syntax: Z←WEEKDAY X

Description:

X: Integer array with its elements in form yymmdd where yy represents years, mm months, and dd days
Z: Index vector giving the days of the week of the dates X. The index of Monday is 1.

Example:

```
      ⎕←V←800101+10000×⍳6
810101 820101 830101 840101 850101 860101
      ⎕←V1←WEEKDAY V
4 5 6 7 2 3
      WDAYS←7 3⍴'MONTUEWEDTHUFRISATSUN'
      (FMTYMD V),' 'BES1 WDAYS[V1;]
01.01.1981 THU
01.01.1982 FRI
01.01.1983 SAT
01.01.1984 SUN
01.01.1985 TUE
01.01.1986 WED
```

Function listing:

```
    ∇ Z←WEEKDAY X;N;A;E;LEAPS;YI;D;⎕IO
[1]  ⍝⍝INDEX OF THE WEEKDAY OF THE DATES X
[2]  ⍝MONDAY=1
[3]   ⎕IO←1
[4]   YI← 78 7
[5]   X←⌈(3⍴100)⊤,X
[6]   N←0,+\ 31 28 ,10⍴5⍴ 31 30
[7]   D←X[3;]+N[X[2;]]+(2<X[2;])^0=4|X[1;]
[8]   E←X[1;]-YI[1]
[9]   LEAPS←((4|E)>4|-YI[1])+LE÷4
[10]  Z←1+7|YI[2]+D+E+LEAPS-2
[11] ⍝840802 11.59
    ∇
```

Name: YD2YMD - Transforms dates in form yyddd into form yymmdd

Syntax: Z←YD2YMD X

Description:

X: Integer array of positive elements. Each element must be given in form yyddd where yy represents years and ddd the day from the beginning of the year. (ddd must be less than 367)
Z: Integer vector with dates in X transformed into form yymmdd where yy represents years, mm months, and dd days.

Example:

```
      ⎕←M←3 2⍴85350+⍳6
85351 85352
85353 85354
85355 85356
      YD2YMD M
851217 851218 851219 851220 851221 851222
```

Function listing:

```
    ∇ Z←YD2YMD X;N;K;L
[1]  ⍝TRANSFORMS DATES IN FORM YYDDD INTO FORM YYMMD
     D
[2]   N←+\0, 31 28 ,9⍴5⍴ 31 30
[3]   L[N+1]←~''⍴L←366⍴0
[4]   X←⌈ 0 1000 ⊤,X
[5]   K←(+\L)[X[2;]-Z←(X[2;]>59)^0=4|X[1;]]
[6]   Z←(10000×X[1;])+(100×K)+X[2;]-N[K]+Z^X[2;]>60
[7]  ⍝840802 12.00
    ∇
```

Name: YMADD - Adds (or substracts) Y months to dates of form yymm

Syntax: Z←X YMADD Y

Description:

X: Array of positive integer elements of form yymm where yy represents years and mm months
Y: Integer array of the same shape as X.
Z: Array X with Y months added to or substracted from the values in X. In similarity with X, the elements of Z are formatted as yymm.

The value(s) of Y should be chosen in a wray that X and Z will be in the same century. This function applies to a scalar and any array accordingly with the primitive function +.

Example:

```
      9812 YMADD 3 10 19
9903 9910 10007
      8012 8410 8808 YMADD 44 ¯2 ¯48
8408 8408 8408
```

Function listing:

```
    ∇ Z←X YMADD Y
[1]  ⍝ADDS (OR SUBSTRACTS) Y MONTHS TO DATES OF FORM
       YYMM
[2]   Z←100⊥1+ 0 12 ⍉(12⊥¯1+ 100 100 ⍉X)+Y
[3]  ⍝840627 14.44
    ∇
```

Name: YMDADDM - Adds (or substracts) months to dates of form yymmdd

Syntax: Z←X YMDADDM Y

Description:

X: Integer array with its elements in form yymmdd where yy represents years, mm months, and dd days
Y: Positive or negative integer scalar or array of the same shape as X
Z: Array X with Y months added to or substracted from the dates in X. The elements of Z are formatted as yymmdd similarly with X.

The value(s) of Y should be chosen so that X and Z will be in the same century. This function needs the auxiliary functions DAYS and YMADD.

Example:

```
      V←831031 750303 460228
      V YMDADDM ¯2
830831 750103 451231
      ⎕←M←(V←CLM V)YMDADDM 3+⍳3
840229
750803
460831
      F←' + 'BES1(⍕CLM 3+⍳3)BES1' MONTHS = '
      (FMTYMD V),F,FMTYMD M
31.10.1983 + 4 MONTHS = 29.02.1984
03.03.1975 + 5 MONTHS = 03.08.1975
28.02.1946 + 6 MONTHS = 31.08.1946
```

Function listing:

```
    ∇ Z←X YMDADDM Y;A;D;L
[1]  ⍝⍝ADDS (OR SUBSTRACTS) MONTHS TO DATES OF FORM
     YYMMDD
[2]  ⍝NEEDS DAYS YMADD
[3]   D←⍴X
[4]   Z← 0 100 ⊤,X
[5]   L←Z[2;]=DAYS Z[1;]
[6]   A←DAYS Z[1;]←Z[1;] YMADD Y
[7]   Z[2;]←AL(Z[2;]×~L)+A×L
[8]   Z←D⍴ 0 100 ⊥Z
[9]  ⍝840802 12.10
    ∇
```

Name: YMDCHK - Checks if dates in X are of form yymmdd

Syntax: Z←YMDCHK X

Description:

X: Numeric array
Z: Scalar 1 if all dates are valid, else 0.
An individual date is valid if it is a positive integer of form yymmdd where years yy do not exceed 99, months mm 12, and days dd 31.

Example:

```
      YMDCHK 991220 010101
1
      V←200601 281301 300215
      (~YMDCHK V)/'PLEASE TRY AGAIN.'
PLEASE TRY AGAIN.
      V[2]←281201
      (~YMDCHK V)/'PLEASE TRY AGAIN.'
```

Function listing:

```
    ∇ Z←YMDCHK X
[1]  ⍝CHECKS IF DATES IN X ARE OF THE FORM YYMMDD
[2]  ⍝RETURNS 1 IF ALL DATES ARE VALID
[3]   X← 0 100 100 ⊤,X
[4]   Z←(X[1;]>0)∧X[1;]<100
[5]   Z←Z∧(X[2;]>0)∧X[2;]≤12
[6]   Z←Z∧(X[3;]>0)∧X[3;]<32
[7]   Z←∧/Z
[8]  ⍝840802 12.19
    ∇
```

Name: YMDTQ - Sequence of consecutive dates from X to Y

Syntax: Z←X YMDTQ Y

Description:

X: Positive integer of form yymmdd where yy
represents years, mm months, and dd days
Y: Positive integer of the form of X.
Y must be larger than X.
Z: Integer vector containing a sequence of
consecutive dates from date X to date Y.
The dates are displayed in form yymmdd.

If the argument dates are not given in chronological order the function returns an empty vector as a result. This function needs the auxiliary functions PIOTA, YMD2YD, YD2YMD, and TQ.

Example:

```
      ⍴⍞←V←850104 YMDTQ 841228

0
      ⍞←V←841229 YMDTQ 850104
841229 841230 841231 850101 850102 850103 850104
      FMTYMD V
29.12.1984
30.12.1984
31.12.1984
01.01.1985
02.01.1985
03.01.1985
04.01.1985
```

Function listing:

```
     ∇ Z←X YMDTO Y;YD1;YD2;V1;V2;E;EV
[1]   ⍝⍝SEQUENCE OF CONSECUTIVE DATES FROM X TO Y
[2]   ⍝NEEDS YMD2YD YD2YMD TO PIOTA
[3]    ⎕IO←1
[4]    Z←⍳0
[5]    YD1←YMD2YD X
[6]    YD2←YMD2YD Y
[7]    →ER⌈⍳YD1>YD2
[8]    V1←⌊YD1÷1000
[9]    V2←⌊YD2÷1000
[10]   →R1⌈⍳V1≠V2
[11]   Z←YD2YMD YD1 TO YD2
[12]   →0
[13] R1:→OK⌈⍳0=E←V2-V1+1
[14]   EV←V1+⍳E
[15]   Z←(1000×EV) PIOTA 365+0=4|EV
[16] OK:V1←YD1 TO(1000×V1)+365+0=4|V1
[17]   V2←(1+1000×V2) TO YD2
[18]   Z←YD2YMD V1,Z,V2
[19] ER:→0
[20] ⍝840802 12.29
     ∇
```

Name: YMD2YD - Transforms dates in form yymmdd into form yyddd

Syntax: Z←YMD2YD X

Description:

X: Integer array of positive elements. Each element must be given in form yymmdd where yy represents years, mm months, and dd days.
Z: Integer vector with dates in X transformed into form yyddd where yy represents years and ddd days from the beginning of the year.

Example:

```
      □←M←3 2⍴(840224+⍳5),840301
840225 840226
840227 840228
840229 840301
      YMD2YD M
84056 84057 84058 84059 84060 84061
```

Function listing:

```
    ∇ Z←YMD2YD X;N
[1]   ⍝TRANSFORMS DATES IN FORM YYMMDD INTO FORM YYDD
      D
[2]   X←⌈(3⍴100)⊤,X
[3]   N←0,+\ 31 28 ,10⍴5⍴ 31 30
[4]   Z←(1000×X[1;])+X[3;]+N[X[2;]]+(2<X[2;])^0=4|X[
      1;]
[5]   ⍝840802 12.31
    ∇
```

Name: YMD2YW - Transforms dates in form yymmdd into form yyww

Syntax: Z←YMD2YW X

Description:

X: Array with positive integer elements in form yymmdd where yy represents years, mm months, and dd days
Z: Integer array in which dates in X are transformed into form yyww where yy represents years and ww the week of the year

Example:

```
      ⎕←M←840000+(10000×⍳3)∘.+15+200×⍳3
850215 850415 850615
860215 860415 860615
870215 870415 870615
      ⎕←N←YMD2YW M
8507 8516 8524 8607 8616 8624 8707 8716 8725
      (3 2 8⍴'YYMMDD: YYWW  : '),⍉M,[1.1](⍴M)⍴N
YYMMDD: 850215 850415 850615
YYWW  :   8507   8516   8524

YYMMDD: 860215 860415 860615
YYWW  :   8607   8616   8624

YYMMDD: 870215 870415 870615
YYWW  :   8707   8716   8725
```

Function listing:

```
    ∇ Z←YMD2YW X;N;A;E;LEAPS;YI;D;⎕IO
[1]  ⍝TRANSFORMS DATES IN FORM YYMMDD INTO FORM YYWW
[2]   ⎕IO←1
[3]   YI← 78 7
[4]   X←⌈(3⍴100)⊤,X
[5]   N←0,+\ 31 28 ,10⍴5⍴ 31 30
[6]   D←X[3;]+N[X[2;]]+(2<X[2;])^0=4|X[1;]
[7]   E←X[1;]-YI[1]
[8]   LEAPS←((4|E)>4|-YI[1])+⌊E÷4
[9]   Z←1+7|YI[2]+E+LEAPS-1
[10]  Z←(100×X[1;])+⌈(D+Z-1+7×Z>4)÷7
[11] ⍝840802 12.33
    ∇
```

Name: YW2YD - First days of weeks X (in form yyww) in form yyddd

Syntax: Z←YW2YD X

Description:

X: Integer array of positive elements. Each element must be given in form yyww where yy represents years and ww the week of the year.
Z: Integer vector returning the first days of weeks given in X. The elements of Z are formatted as yyddd where yy represents years and ddd the day of the year.

Example:

```
      ⎕←V←8600+20+⍳5
8621 8622 8623 8624 8625
      YD2YMD ⎕←YW2YD V
86139 86146 86153 86160 86167
860519 860526 860602 860609 860616
```

Function listing:

```
    ∇ Z←YW2YD X;E;LEAPS;YI;⎕IO
[1]  ⍝FIRST DAYS OF WEEKS X (IN FORM YYWW) IN FORM Y
     YDDD
[2]  ⎕IO←1
[3]  YI← 78 7
[4]  X← 100 100 ⊤,X
[5]  E←X[1;]-YI[1]
[6]  LEAPS←((4|E)>4|-YI[1])+⌊E÷4
[7]  Z←1+7|YI[2]+E+LEAPS-1
[8]  Z←(1000×X[1;])+1⌈(7×X[2;]-Z<5)-Z-2
[9]  ⍝840802 12.36
    ∇
```

Name: INPC - Displays prompt X appended by KP.
Returns the user's character input

Syntax: Z←INPC X

Description:

X: Character scalar or vector.
Z: Displays prompt X appended by a global variable KP (character scalar or vector). The user's character input is returned in Z.

KP is usually a colon or a question mark.

Example:

```
      KP←':'
      V←INPC'YOUR ANSWER'
YOUR ANSWER : I TAKE IT
      V
I TAKE IT
```

Function listing:

```
     ∇ Z←INPC X;A
[1]  ⍝⍝DISPLAYS PROMPT X APPENDED BY KP. RETURNS THE
       USER'S CHARACTER INPUT
[2]  ⍝NEEDS GLOBAL VARIABLE KP
[3]   A←⍴⎕←X,' ',KP,' '
[4]   Z←A↓⎕
[5]  ⍝840802 14.23
     ∇
```

Name: INPN - Displays prompt X appended by KP.
Executes the user 's numeric input

Syntax: Z←INPN X

Description:

X: Character scalar or vector
Z: Displays prompt X appended by a global variable KP (character scalar or vector). The user's numeric input is executed and returned in Z.

KP is usually a colon or a question mark.

Example:

```
      KP←'?'
      V←INPN'WHICH YEARS'
WHICH YEARS ? 1980+⍳7
      V
1981 1982 1983 1984 1985 1986 1987
```

Function listing:

```
    ▽ Z←INPN X;I
[1]  ⍝⍝DISPLAYS PROMPT X APPENDED BY KP. EXECUTES TH
      E USER'S CHARACTER INPUT
[2]  ⍝NEEDS GLOBAL VARIABLE  KP
[3]   I←⍴⍞←X,' ',KP,' '
[4]   →(' '^.=I←I↓⍞)/⍴Z←⍳0
[5]   Z←⍎I
[6]  ⍝840802 14.25
    ▽
```

Name: YES - Displays prompt X appended by KP. Returns 1 if user replies positively

Syntax: Z←YES X

Description:

X: Character vector or scalar
Z: Function displays prompt X appended by a global variable KP (character vector or scalar). If user replies positively Z will be 1, else 0. An answer is positive if the first non-blank character is y,Y,K, or K (or APL ↑ or ').

It is easy to include other characters to the ones that are considered positive. Just add these characters to the sequence within the quotation marks in line 4. KP is usually a colon or a question mark.

Example:

```
      KP←':'
      (YES'TO CONTINUE WRITE Y')/'NEXT STEP'
TO CONTINUE WRITE Y : YEAH
NEXT STEP
```

Function listing:

```
    ▽ Z←YES X
[1]  ⍝⍝DISPLAYS PROMPT X APPENDED BY KP. RETURNS 1 I
      F USER REPLIES POSITIVELY
[2]  ⍝NEEDS GLOBAL VARIABLE KP
[3]   Z←Z↓⍨1,0/Z←⍴⎕←X,' ',KP,' '
[4]   Z←∨/'Y↑K'''=1↑(∨\Z≠' ')/Z
[5]  ⍝840802 14.26
    ▽
```

Name: FNNAME - Names of the functions when X contains the headers

Syntax: Z←FNNAME X

Description:

X: Character scalar, vector, or matrix
Z: Character vector or matrix the rows of which return the names of the functions whose headers are given in X. The corresponding rows of global variable TYPE indicate the type of a result and the number of arguments. 0 stands for no result and 1 for a result.

The rows of X must not contain extra blanks between the individual components of each header.

Example:

```
      FNNAME'Z←X UPON Y'
      TYPE
1 2
      ⎕←M←(1 40↑⎕CR'DTB')UPON 1 40↑⎕CR'PRTINIT'
Z←DTB X;⎕IO
PRTINIT;A
      FNNAME M
DTB
PRTINIT
      TYPE
1 1
0 0
```

Function listing:

```
    ∇ Z←FNNAME X;A;L;P;M;⎕IO
[1]   ⍝⍝NAMES OF THE FUNCTIONS WHEN X CONTAINS THE HE
      ADERS
[2]   ⍝GLOBAL TYPE ↔ RESULT, NUMBER OF ARGUMENTS
[3]   ⍝X MUST NOT CONTAIN EXTRA BLANKS
[4]    A←¯1+(L←X=' ')⊥⎕IO←1
[5]    TYPE←(∨/B←X='←'),[1.1],(+/L)-A
[6]    P←((TYPE[;1]^TYPE[;2]≠2)×1++/^\~B)+(TYPE[;2]=2
       )×1++/^\L←~L∨X=';'
[7]    L←P⌽L
[8]    X←P⌽X
[9]    M←⌈/P←+/^\L
[10]   X←(Z,M)↑P⌽(((Z←¯1↓⍴X),M)⍴' '),X
[11]   Z←(X+.=' ')⌽X
[12]  ⍝840802 14.29
    ∇
```

Name: INFUNCTION - Finds string X in the function Y

Syntax: Z←X INFUNCTION Y

Description:

X: Character string (scalar or vector)
Y: Function name in a character string (scalar or vector)
Z: Two-row matrix with the first row containing the lines and the second row the locations of the string X in the function Y

If the function is not found or if the string is not found in the function, the result is an empty matrix. This function uses the auxiliary function INVECTOR.

Example:

```
      '⎕IO' INFUNCTION 'INFUNCTION'
   4 RANK ERROR
INFUNCTION[5] Z←(⍴Y)⊤X INVECTOR,Y←⎕CR Y
                                   ^
```

Function listing:

```
    ∇ Z←X INFUNCTION Y;⎕IO
[1]  ⍝⍝FINDS STRING X IN THE FUNCTION Y
[2]  ⍝Z[1;] CONTAINS THE LINES AND Z[2;] THE LOCATIO
      NS
[3]  ⍝NEEDS INVECTOR
[4]   ⎕IO←0
[5]   Z←(⍴Y)⊤X INVECTOR,Y←⎕CR Y
[6]  ⍝840802 08.02
    ∇
```

Name: INWS - Finds string Y in the workspace

Syntax: X INWS Y

Description:

X: Non-negative integer scalar or vector
Y: Character scalar or vector

Seeks for the string Y in the workspace. Displays the functions, lines and locations. Displays the string Y, X[1] characters before Y, and X[2] characters after Y if X∨.≠0. A scalar argument X is reshaped to 2⍴X.
This function uses the auxiliary functions INVECTOR and ALFASORT.

Example:

```
      8 10 INWS 'DABA'

DABA

  0  2          Z←DABA A;L

PAGEMOD

 24 85   NC XNCD DABA PRTUTILGP

XNC

  3  7   1⍴NEEDS DABA
  5  9   0×⍳0∈⍴Y←DABA(X,L(X/⍴Y)
       0 INWS 'RIOTA'

DOKU

 47 43

RIOTA1

  0  4

RIOTA2

  0  4
```

Function listing:

```
     ∇ X INWS Y;C;D;E;F;⎕IO
[1]   ⍝⍝FINDS STRING Y IN THE WORKSPACE
[2]   ⍝DISPLAYS THE FUNCTIONS, LINES AND LOCATIONS
[3]   ⍝PRINTS X[1] CHARACTERS BEFORE Y, THE STRING Y,
        AND X[2] CHARACTERS AFTER Y (IF X∨.≠0)
[4]   ⍝NEEDS INVECTOR ALFASORT
[5]    ⎕IO←1
[6]    X←(X∨.≠0)/(-X[1]+1)+⍳(⍴,Y)++/X←2⍴X
[7]    D← 3 9 ⍴'INVECTOR ALFASORT INWS       '
[8]    C←ALFASORT(∧/C[;⍳9]∨.≠⍪D)≠C←(⎕NL 3),' '
[9]   R1:→0×⍳0≥⍴C
[10]   →R2⌈⍳0=⍴D←Y INVECTOR E←,F←⎕CR C[1;]
[11]   ''
[12]   C[1;]
[13]   ''
[14]   (3 0 ⍕⍪(⍴E)⍪D-1),(((⍴D),3)⍴' '),E[1⌈(⍴E←' ',E,
       ' ')⌊1+D∘.+X]
[15] R2:→R1,0⍴C← 1 0 ↓C
[16] ⍝840802 08.07
     ∇
```

Name: NEWFNS - Returns functions that contain a time stamp of form ⍺YYMMDDHHMM

Syntax: Z←X NEWFNS Y

Description:

X: Character matrix with names of functions on its rows. Can be an empty vector or a scalar, too.
Y: Positive integer of form yymmdd[hhmm] where yy represents years, mm months, dd days, optional hh hours, and optional mm minutes
Z: Character matrix returning the functions that contain a time stamp of form Y as the last comment line and the time stamp is ≥ Y. If X is a character matrix, only the functions given in X are checked. Otherwise, all the functions in the workspace are examined.

This function yields another result in a global variable TS. TS is an integer vector giving the time stamps of the functions returned in Z.

Example:

```
      (' 'LISTINWS NEWFNS DTB') NEWFNS 840801
NEWFNS
      TS
8408030729
      '' NEWFNS TODAY
INTRO
NUMEROI
PRTINIT
SIS
TULOSTA
      TS
8412031012 8412031250 8412031247 8412031249 8412031308
```

Function listing:

```
     ∇ Z←X NEWFNS Y;X;K;I;D;C
[1]   ⍝RETURNS FUNCTIONS THAT CONTAIN A TIME STAMP OF
        FORM ⍝YYMMDDHHMM
[2]   ⍝AS THE LAST COMMENT LINE AND THE TIME STAMP IS
        ≥ Y
[3]   ⍝RETURNS A GLOBAL VARIABLE IS WITH THE TIME STA
       MPS
[4]   ⍝X CONTAINS THE NAMES OF THE FUNCTIONS OR '' FO
       R ALL IN THE WS
[5]   ⍝Y IS OF FORM DATE (YYMMDD) [TIME (HHMM)]
[6]    Y←10000⊥2↑Y
[7]    ⊥(2≠⍴⍴X)/'X←⎕NL 3'
[8]    IS←K←0/I←⎕IO
[9]   R1:→R2⌈⍳0⍷D←1⍴⍴C←⎕CR X[I;]
[10]   →R2⌈⍳0=D←×/¯1↑(C[;⎕IO]='⍝')/⍳D
[11]   →R2⌈⍳~∧/(C←15↑C[D;])∈'⍝1234567890. '
[12]   C←10000⊥⊥(~C∈'⍝.')/C
[13]   K←K,(Z←C≥Y)/I
[14]   IS←IS,Z/C
[15]  R2:→R1⌈⍳(⎕IO+1↑⍴X)>I←I+1
[16]   Z←X[K;]
[17]   Z←Z[C←⍋256⊥⍉⎕AV⍳Z[;⍳10⌊1↓⍴Z];]
[18]   IS←IS[C]
[19]  ⍝840803 07.29
     ∇
```